Noname manuscript No.
(will be inserted by the editor)

A Few-Shot Learning Framework for Planar Pushing of Unknown Objects

Received: date / Accepted: date

Abstract Robot planar pushing is one of the primitive elements of non-prehensile manipulation skills, and has been widely studied as an alternative solution to complex manipulation tasks. To transfer this skill to novel objects, reasoning the pushing effect on object motion is important for selecting proper contact locations and pushing directions. However, complex contact conditions and unknown physical properties of the object cause difficulties in reasoning.

In this work, firstly, we present a new large planar pushing dataset that contains a wide range of simulated objects, and a novel representation for pushing primitives for the data-driven prediction model. Secondly, we propose a computation efficient planning method that employs a heuristic to reduce the possibility of making sliding contact between the pusher and the object. The prediction model and planning method were evaluated both in simulation and real experimental settings. The results show that the prediction model purely trained using our simulation dataset is capable of predicting real object motions accurately. The push planning method effectively reduces the number of pushes required to move unknown real objects to target positions.

Keywords Planar Pushing · Path Planning · Non-prehensile Manipulation · Data-driven Automation

1 Introduction

Planar pushing gives a simple yet efficient way to change the state of an object. There are many studies to apply pushing in complex tasks such as object placement [9], object singulation [15], or robotic grasping [7, 10, 12]. However, it is difficult for robots to efficiently apply this skill to the scenario where the environment and the manipulated object are both unknown. To apply pushing manipulation to novel objects, a prediction model is needed to reason the pushing effect on object motion. Recently, Stuber *et al.* [42] reviewed the methods for predicting the motion of the pushed object with machine learning-based approaches that benefit from their enhanced performance on modeling complex object motions with fewer assumptions. However, there exist several challenges with machine learning models to generalize to novel object behaviors. Firstly, it is commonly recognized that the scale and quality of the dataset play an important role in dealing with the generalization issue. But the large scale and diverse datasets are difficult to get, as collecting data using a real robot is extremely expensive and time-consuming. Until now, several pushing datasets have been presented [5, 51]. However, the datasets contain a limited number of objects that are mostly similar in shape and size. In practice, an infinite number of real-world objects and their unknown physical properties make machine learning models difficult to generalize. Other than that, an appropriate representation for a machine learning model is also required in order to scale to objects with various shapes, sizes, and parameters [5].

This work extends our previous work in [24] to deal with planar pushing in two aspects: push affordance prediction and push planning. The term push affordance, borrowed from Kloss *et al.* [30], refers to the resultant object motion of robot pushing. The push planning method aims to push a novel object to the target position with the help of the predicted push affordances.

There are some studies showing that the knowledge learned from simulation can be transferred to real settings with (or without) a small amount of real data [6, 30, 32, 34, 49]. Depierre *et al.* [11] and Byravan *et al.* [6] presented a large-scale dataset collected in a simulation environment stressing the advantage in both scale and diversity. Motivated by these works, we release a large-scale, contact-rich pushing dataset called **SimPush** to deal with the push affordance pre-

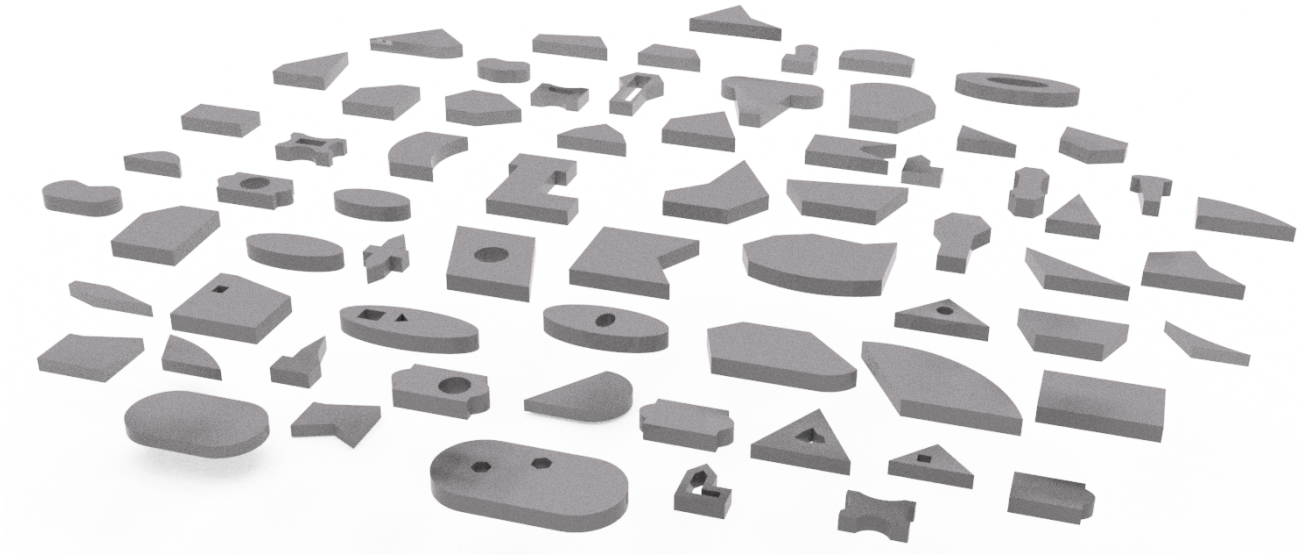

Fig. 1 SimPush: Large-Scale Planar Pushing Dataset available at https://github.com//SimPush

diction problem. The **SimPush** dataset contains 69 objects (depicted in Fig. 1) with diverse dimensions and shapes that appear either convex or concave. A vast variety of physical properties are considered; surface and contact frictions, the center of mass (CoM), mass, and moment of inertia of the object. These properties are arranged in different combinations to change the motion of the object pushed at various contact points in different directions leading to more than 2 million pushes in total. Then, we adopt a few-shot learning model to predict push affordances. The few-shot learning model leverages a small set of pushing priors aiming to infer pushing affordance for other pushing actions. This model can help deal with an infinite number of real-world objects using a limited dataset. Finally, we propose a compact method for representing pushing primitives to preserve the spatial relationship between the object and the pusher. It helps the learning model encode the relation between object state changes and the applied action. We trained the few-shot learning model integrated with the proposed representation method using the **SimPush** dataset, and then empirically evaluate the model performance in simulation and a real setting. The results show that the proposed method not only outperforms existing models but also demonstrates the robust prediction of unknown object motions.

For push planning task, we use the trained push affordance model to predict the push affordances for a set of sampled pushing actions. After that, we propose to employ a heuristic method aiming to reduce the possibility of making sliding contact between the pusher and the object. Given a specific task, we design a novel cost function and a small set of pre-selected efficient pushing actions to evaluate all pushing actions. The efficiency of the planning method is measured by the following two aspects; the first one is the accumulated movement of object, since larger accumulated movements are most likely a result of sliding contacts between the pusher and the object during pushing. The second one is the number of pushing steps. The more pushing steps executed, the less efficient the planning is. Through extensive simulation and real robot experiments, our method is demonstrated to be robust against changes in the object's pose, size, and shape.

To summarize, the main contributions of this work are:

- A large-scale and diverse planar pushing dataset that contains convex and concave objects with different physical properties.
- A push affordance prediction model and a compact state representation for planar pushing.
- A push planning method integrating a push affordance prediction model to translate the novel object to the desired position efficiently.
- Extensive evaluation and comparison of the proposed method by conducting a series of experiments in both simulation and real settings.

2 RELATED WORK

We categorized push-related research under four different categories; Dataset, Model Identification, Prediction Model for Planar Object Motion, and Planning.

2.1 Dataset

Learning-based methods are capable of dealing with unseen objects using various datasets [5, 19, 33, 39, 47] for train-

ing. Yu *et al.* [51] made great effort on collecting a planar pushing dataset using a high fidelity real robot system. They collected pushing data using 11 objects over 4 different surfaces with different pushing velocities. However, the number of the objects was limited and the influence of the physical properties of the objects to the resultant motion was not fully studied. Recently a pushing dataset called Ommipush [5] was presented. It contains objects formed by combining 4 different magnet sides and using extra mass to change object weight and CoM. However, objects generated using magnets lacks a variety in shapes due to side-sharing and limited physical properties. Compared with the aforementioned dataset, our **SimPush** dataset is characterized by the large-scale diverse physical responses to pushing action. There are other push datasets for different purposes. Finn *et al.* [18] proposed a pushing dataset that contains 57 thousands pushes to directly model the pixel motion in the image frame. Eitel *et al.* [17] proposed a pushing dataset for object singulation task. Pulkit *et al.* [1] presented a pushing dataset collected in a self-supervised way for learning intuitive physics. However, these datasets only contain visual and robot pushing information and the physical properties of the object are not considered.

2.2 Model Identification

Different studies have been conducted to predict object motion incorporating system identification methods [21, 30, 41, 44, 46, 48, 52] or to estimate system parameters through observing object motion [2, 3, 32, 38, 39, 46]. For the former studies, the object physical properties are usually either explicitly estimated [30, 46] or implicitly represented by neural networks [21, 44, 48, 52]. Specifically, Wu *et al.* [46] fed explicitly estimated physical parameters as input to an analytical model of physical system to estimate object motion. In [30], the Kalman filter was used to estimate object physical properties (such as the CoM, friction, mass, and others). However, the prediction accuracy was related to the quality of the estimation that is likely to be intractable for the objects having complicated contact phenomena. In addition, Song *et al*. [41] proposed to learn the coupled mass-friction parameters through minimizing the simulation-reality gap. This method needs to keep a set of hypothesized mass and friction models, whereby the local optimality and the approximation made in the simulator limit its capability. On the other hand, Fragkiadaki *et al.* [21] predicted ball motion under a pushing force by leveraging the most recent glimpse of the object-centered image patches. Wang *et al.* [44] proposed to collect the tactile feedback data when executing predefined motion to implicitly encode physical properties of novel objects. For the latter studies, Allevato *et al.* [2, 3] used a neural network to tune the parameters of the physics engine based on the difference in observation from the real object motion, leading to a more accurate simulation. However, it is limited only to known objects. Mavrakis *et al.* [38] proposed to estimate inertial properties for the object by planar pushing and data-driven models. In our research, instead of identifying object physical properties, we propose to use the few-shot learning model leveraging a few pushing priors to predict object motion for pushing action.

2.3 Forward Model for Planar Object Motion

Mason [37] proposed an analytic model for quasi-static planar pushing. Goyal *et al*. [26] introduced the limit surface reasoning the frictional forces with object motion. Kloss *et al*. [31] proposed a hybrid model combining a data-driven deep learning model and the analytical model adopted in [36]. These studies relied on the assumption that friction is known. Several studies [14, 16, 18, 43] put predicting action effects into the video prediction scenario with the self-supervised learning method. Hermans *et al*. [28] proposed a regression model to evaluate the pushing effects for a set of contact points. On the other hand, Li *et al*. [34] proposed a data-driven method utilizing the experience of push interactions with novel objects to implicitly learn a forward model encoding the relationship between the action and object state. However, the accuracy of the learned model was not investigated. In this work, we adapt their Push-Net model to explicitly learn to predict the state of the object pushed. Bauza *et al*. [5] and Goo *et al*. [25] used Attentive Neural Process (ANP) [29] to learn the object dynamics without considering object shapes. They predicted the action effect on the entire state of the object, which might not be easily obtained in real scenarios.

2.4 Planning

Some studies focus on learning an inverse model which aims to find the proper action for achieving the target without reasoning the action effects [1, 22, 23, 53]. Specifically, Zeng *et al*. [53] proposed the transporter network to infer object displacement and robot action through finding the correspondence of the deep features. However, dealing with out-of-distributed objects might be difficult because of the unknown physical properties. Apart from that, Lin *et al*. [35] and Arruda*et al*. [4] employ the model predictive path integral approach to optimize the pushing action. The cross-entropy method is also intensively used in [15, 45, 45, 50]. These methods need roll-out simulation from current state which is computationally expensive. Cosgun *et al*. [9] utilized planar pushing to create space for placing objects on a cluttered surface. Specifically, they propose a heuristic, which favors less overlap with the object to be placed, to simplify the push plans. Cosgun *et al*. [9] and Dogar *et al*. [12]

employ a simulator to predict pushing effect on the objects that exist in the scene which are assumed to be known. In contrast, our push planning method deals with unknown objects. Florence *et al*. [20] proposed to build the policy model upon the visual correspondence model to speed up training. This method seems promising but re-training the policy model is needed to push a new object. In this work, different from the aforementioned methods, we propose a computation-efficient planning method that integrates the predicted push affordances to reduce the possibility of sliding contact between the pusher and the object, leading to efficient sequences of pushing novel objects to the target positions.

3 PUSHING SIMULATION DATASET: SIMPUSH

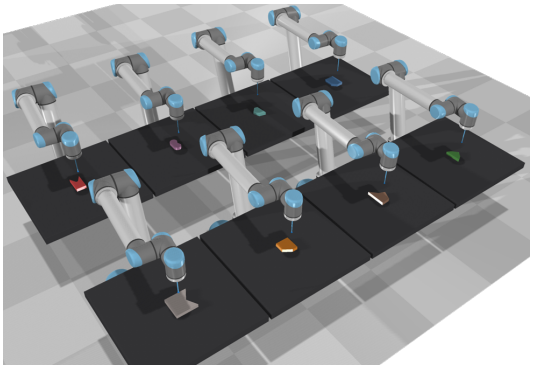

Fig. 2 Planar pushing simulation environment developed in CoppeliaSim.

3.1 Simulation Environment

The simulation environment was created with CoppeliaSim [40] as shown in Fig. 2. A spherical pusher with a diameter of $0.95cm$ is attached to a cylinder with a length of $20cm$ to be the pusher, while a 6-DOF articulated manipulator holds and controls the position of the pusher. We created 5 flat floor surfaces with various coefficients of friction. Each object is pushed across the floor. There are mainly four physics engines available: Bullet, ODE, Vortex, and Newton. We compared the performance of the engines in the aspect of stability and repeatability. For the stability test, we assigned different physical properties to the same objects. The Vortex engine has proven the most robust against the assigned physical properties. For the repeatability test, we applied the same pushing action to the objects a number of times with low linear velocity causing a change in the state of the objects. We found that both the mean and standard deviation of the change were minimum when using the Vortex engine. Therefore, we choose Vortex Studio's physics engine [8] to simulate the dynamic interaction between the pusher and diverse objects.

3.2 Objects

We designed 69 objects that come in a variety of shapes, either convex or concave. The objects are diverse in size, ranging from $3.5cm \times 6cm$ to $20cm \times 17cm$. For each object, we make 40 different combinations of physical properties, including the contact friction of the side surface (μ_c), mass (M), inertia (I, calculated by scaling the default inertia tensor given by the simulator), and CoM. The range of each physical properties are shown in Table 1. The CoM of the object is defined by taking the following steps: First, the object mask is segmented from an image captured using a depth camera. Since the CoM of the object is located inside the convex hull of the object mask, we randomly sample a position inside the convex hull as the CoM location. Then, we specify the CoM location relative to the object frame at the centroid based on the transformation matrix between the camera and the object frames.

We then assign specific physical properties and 5 different frictional surfaces (μ_s) to each object. Finally, we have 200 different contact dynamics for each object.

Table 1 Summary of parameters used in SimPush

Surface friction coefficients (μ_s)	0.2, 0.4, 0.6, 0.8, 1.0
Contact friction coefficients (μ_c)	0.5, 1.0
Number of center of mass for each object	10
Ratio range of object moment of inertia	[0.01, 100]
Range of object mass	[50g, 400g]
Number of pushes for each object	180
Number of shapes	69

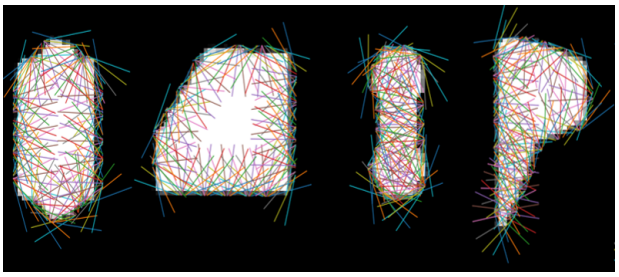

Fig. 3 Pushing samples for different shapes. For each shape, 17-18 contact points are uniformly sampled across the object perimeter , each of which has 10 different push directions. The line with different color represents different pushing direction *w.r.t.* the contact normal direction.

Algorithm 1 Data Collection Scheme

procedure COLLECT($object, \mu_s, CoM, \mu_c, I, M$)
  Set surface friction μ_s
  Load $object$, determine initial object state O_{init}, set CoMs, μ_c, inertia I, and mass M.
  Sample m points $\{cp_i\}_{i=1,\dots m}$ uniformly across the object perimeter.
  Calculate n push directions $\{d^i_j\}_{j=1,\dots,n}$ in range of $[-75^\circ, 75^\circ]$ *w.r.t.* the normal of each contact point $\{cp_i\}_{i=1,\dots m}$.
  cache = [ ]
  cache.append(O_{init})
  for cp_i in $\{cp_i\}_{i=1,\dots m}$ **do**
    for d^i_j in $\{d^i_j\}_{j=1,\dots,n}$ **do**
      Reset object to O_{init}.
      $\epsilon \leftarrow$Random-Sample() ▷ small perturbation
      Move pusher to $cp_i \pm \epsilon$.
      Push object $3cm$ along d^i_j.
      Get object state O_{ij}
      cache.append($cp_i \pm \epsilon, d^i_j, O_{ij}$)
  return cache

For each object with combinations of physical properties and μ_s, push data are collected as given in Algorithm 1. Each run returns around 180 pushes. In this work, we assume quasi-static interactions, where the inertial force is in the suborder of the frictional forces. The velocity is controlled by the displacement per time step. We set this displacement to be 0.3 mm with the time step of 50 ms, yielding a pushing velocity of 6 mm/s. We assume that this velocity is small enough to satisfy the quasi-static condition. In order to verify if this parameter setting produces both reliable and repeatable object motions, we chose a rectangular object and selected 18 pushing points uniformly along the object perimeter. The robot pushed each point 100 times with the same set of randomly selected directions with 3 cm forward. We observed that the resultant position and orientation of the object were all close enough for each pushing point. The start location of the pusher can be either in contact or not in contact with the object. Fig. 3 shows the pushing samples on four example objects in **SimPush**. We repeat this procedure for each object until the simulation goes through all the combinations of physical properties and surfaces. We finally create more than 2 million pushes. For each push sample, the following information is recorded.

- **RGB-D Image**: A camera mounted on top of the table operates in orthographic projection mode. We obtain the corresponding mask image by re-projecting the point cloud to the image plane with 224×224 resolution.
- **CoM**: We record the CoM of the object before and after pushing. Note that all the objects have the same thickness, and the locations of CoM are given in the image frame in the xy-plane. One-half the thickness is assigned to the z coordinate of CoMs of the objects.
- **Action**: Actions are represented by the starting and terminating position of the pusher in the image frame.
- **Object Pose**: We record the pose of the object before and after pushing.
- **Properties**: Mass, inertia, contact friction, and surface friction are stored for the implementation of the baseline model.

In Table 2, **SimPush** is compared with the existing datasets.

Table 2 Comparison with existing push datasets

Dataset	objects	surfaces	pushes	Platform	Size
SimPush	**2760**	5	180	Simulation	∼**2M**
Omnipush [5]	250	1	250	Real	∼63K
Yu [51]	11	4	6000	Real	∼264K

4 METHODS

We formulate the push affordance prediction problem as follows: given m pushing priors $\{A_i, \Delta O_i\}_{i=1,\dots,m}$ and n test data $\{A_j\}_{j=1,\dots,n}$, A is the pushing actions, and ΔO is the changes in object state represented by $\Delta x, \Delta y, \Delta\theta$. We assume all the object poses in pushing priors and tests before pushing are the same. The problem is how to efficiently incorporate the pushing priors to predict the ΔO for test examples. Given the target and initial object pose, and a set of push actions with the corresponding push affordances, the push planning problem is how to iteratively select the push action based on the target direction and the push affordances to optimize the number of required pushing. In this section, firstly we introduce the representation for pushing and the pre-processing for input. Then we introduce the few-shot learning model. Finally, we explain the method used in push planning.

4.1 Pushing Primitive Representation

4.1.1 Action maps

The way of describing pushing primitives plays an essential role in modeling the push affordance. Pushing primitives can be described by the pusher's starting and terminating positions [25,30,31,34], commonly represented by a 4×1 vector $[x_s, y_s, x_t, y_t]^\top$ in which the first two dimensions are for the starting position and the other two dimensions for the terminating position.

However, we found that this simple representation tends to make the learning model to be overfitting the training dataset. This might be due to the fact that the total number of object shapes is still limited even though the number of collected pushing actions in the dataset is large. In this work, we propose a representation for the pushing action

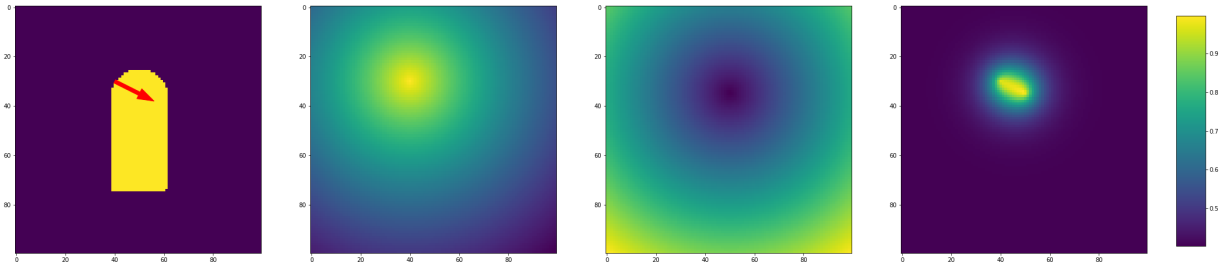

Fig. 4 Action maps: The first one is the mask image and the applied pushing action, the middle two images are the action maps generated based on start pushing position and end pushing position respectively. The last image is to visualize the difference between action maps.

that provides better generalization capability to novel object shapes. Specifically, we use the distance transform formulation defined by Eqs. 1 through 3 to create two images with the same resolution as mask images called action maps. The term c in Eqs. 1 and 2 is the normalization term that depends on the size of the image used. The first action map is created based on the pushing start position; the closer to the starting position, the higher pixel value assigned. The second action map is created based on the pushing end position; the closer to the ending position, the lower pixel value assigned. The first action map guides the model to focus on the geometric features at the contact point. The pushing direction and magnitude can be inferred from the difference between the two action maps. Fig. 4 shows an example of the generated action maps and the visualization between those action maps.

$$s(x, y, x_s, y_s) = e^{-\frac{d(x,y,x_s,y_s)}{c}} \tag{1}$$

$$t(x, y, x_t, y_t) = \frac{d(x, y, x_t, y_t)}{c} \tag{2}$$

$$d(x, y, x_p, y_p) = \sqrt{(x - x_p)^2 + (y - y_p)^2} \tag{3}$$

4.1.2 Push Embedding Model

To maintain the spatial relation between the applied pushing action and object state, and to help the learning model focus on the local contact feature and pushing, we stack the object mask and action maps along the channel axis to be the one of the inputs to pushing embedding model. As mentioned at the beginning of this section, the pushing priors contain object mask, applied actions and the resultant object motions. The pushing tests only contain object mask and the applied actions. There is a slight difference in embedding between pushing priors and tests. The proposed pushing embedding model is shown on the left in Fig. 5. For embedding pushing priors, a Convolutional Neural Network (CNN) takes the stacked object mask and action maps as input and outputs f_d, while ΔO is reshaped to the same dimension as f_d by the Fully Connected Network (FCN). After that, f_e is obtained by concatenating them together. Notably, only a CNN model is needed to embed pushing tests to f_d.

For the CNN part, we use 5 pre-trained layers of ResNet50 [27] to construct the base structure. On top of pre-trained layers, we build a 1×1 2-D convolution layer and one FCN. CNN outputs a 256-D feature vector f_d. FCN for reshaping ΔO is constructed by two layers with the same dimension of 256 with ReLU activation function.

4.2 Proposed Learning Model

We develop the push affordance prediction module based on ANP [29] to model the causality between the applied pushing action and the resultant object motion. Fig. 5 shows the proposed push affordance prediction model (middle and right). The main motivation behind using ANP is that, firstly, ANP is a powerful model for regression tasks. It can predict an arbitrary number of test data based on an arbitrary number of priors while keeping computation complexity linear *w.r.t.* the number of priors. Then, it has two self-attention modules to boost the representation of the priors. Two self-attention modules boost each isolated prior by integrating the attentions calculated with all the priors, but the second one performs the mean operation upon the output to obtain a permutation-invariant representation. As a result, the first attention module outputs a representative feature for each prior, while the second attention module only outputs a single representative feature given all priors. In addition, ANP has a cross-attention module to imitate the functionality of the kernel in the Gaussian process to compute the similarity between tests and priors to improve fitting performance. This is also reasonable in our application scenario, since similar pushing actions should have similar resultant object motions. Finally, a multi-layer perceptron (MLP) is used to predict the corresponding label for test data. In our model architecture, there are also two self-attention modules and one cross-attention module with the same structure as ANP. However, we replace the MLP with an attention module named as cascaded residual attention. The cascaded resid-

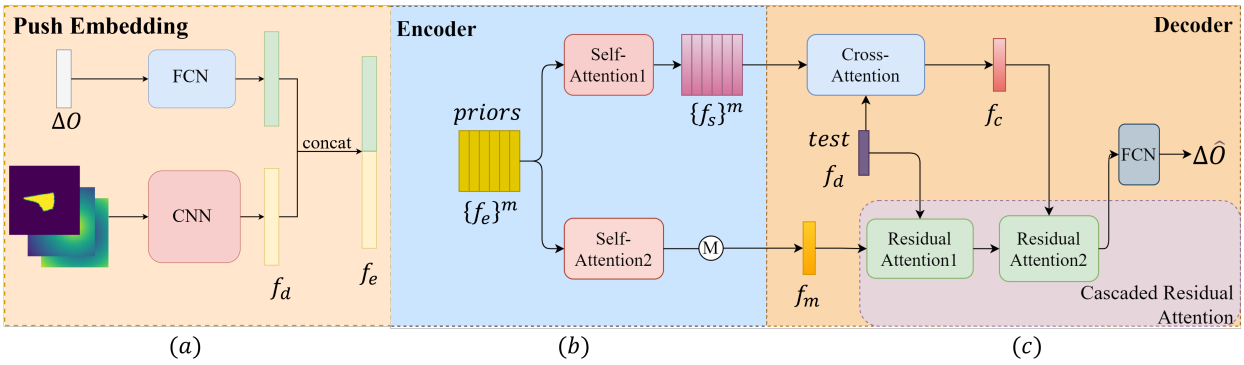

Fig. 5 Proposed push embedding model (a) and encoder-decoder learning model (b and c). In push embedding, CNN takes the stacked object mask and action maps as input and encodes the spatial relations between the object state and action. FCN projects ΔO to the same dimensional space as f_d. Residual attention module is utilized to selectively combine them to output f_e.

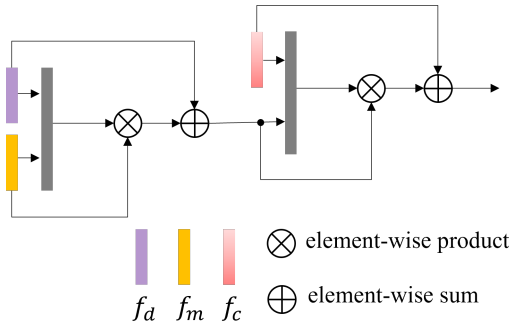

Fig. 6 Cascaded residual attention module. the long gray blocks are the fully connected layers.

ual attention selectively combines different source inputs to enhance inference capability.

The cascaded residual attention model consists of two residual attention modules. For each residual attention, it takes two feature vectors of the same length to output a representative feature vector of two inputs with the same size. The process can be represented by Eqs. 4 and 5.

$$F_{attn}(f_{in1}, f_{in2}) = tanh(concat(f_{in1}, f_{in2})W^T + b) \quad (4)$$

$$f_{out} = F_{attn}(f_{in1}, f_{in2}) \cdot f_{in2} + f_{in1}, \quad (5)$$

where $tanh$ is the activation function. The cascaded residual attention model in decoder is shown in Fig. 6

Let us assume that there are m pushing priors and only one test. During encoding, the pushing priors are fed into the proposed push embedding module and self-attentions to get $\{f_s\}^m$ and f_m, where f_m is the element-wise mean of the output of the second self-attention module. During decoding, the test is embedded to f_d, then the cross-attention module takes f_d and $\{f_s\}^m$ as input to output f_c. After that, f_c, f_d, f_m are fed into a cascaded residual attention module and FCN to output the predicted object motion for the test.

In order to measure the uncertainty of the prediction, we design the FCN in decoder to predict both object motion and prediction uncertainty by outputting mean and standard deviation. Since the standard deviation must be a non-negative real number, we add a ReLU layer after the linear layer for the uncertainty. We use non-negative log-likelihood as the loss function to minimize the prediction error.

4.3 Planning

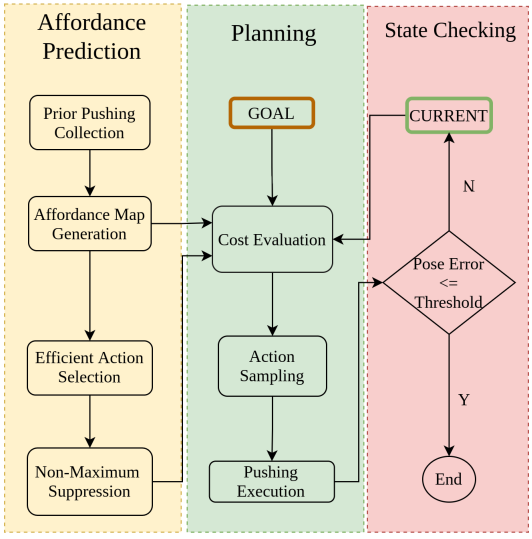

Fig. 7 Pushing planning flowchart given the object's initial and target poses. A few prior pushes are collected for encoding the object dynamics. Then, the push affordance map is generated by the trained model to predict the effect of the representative actions. After that, the representative actions are ranked based on the translating effect. The top ranked actions are selected as candidates of the efficient actions. Finally, the non-maximum suppression method is utilized to get the efficient actions. The proposed cost function quantifies each representative actions. Then we sample an action among the top ranked actions considering uncertainties associated. This procedure will continue until the pose error between the current and target state meets the given criterion.

Planning is to push the object from its initial pose to the target within pre-specified steps as shown in Fig. 7. Before starting the planning process, several pushes are executed to interact with the object and these interactions are fed into the few-shot learning model to encode object dynamics. The push affordance map specifies the object pose as well as the effects associated with the representative actions. It is calculated only one time and will be reused multiple times until the planning task ends. In the phase of push affordance map generation, we sample multiple points on the outline of the object mask and take 5 directions *w.r.t.* the surface normal ($0^{\circ}, \pm 30^{\circ}, \pm 60^{\circ}$) as representative actions $\{a_r\}^n$ similar to [30]. Then the push affordance map is generated using the prediction model to predict the push affordance for all representative actions.

In the phase of Efficient Action Selection in Fig. 7, we select a set of candidates for the efficient actions from the representative actions. Those candidates efficiently translate the object, causing small changes in the object's orientation. However, in practice, there are many candidates similar in contact position as well as pushing direction, which implies that we can make some calculations unnecessary. Furthermore, the candidates with a large amount of uncertainty should not be included in the efficient action set. We therefore sort the action candidates based on the predicted uncertainties. Then using Non-Maximum Suppression, we filter out candidates similar in contact position or ones with large uncertainties. The remained actions are referred to as the efficient actions denoted by $\{a^*\}^m$.

In the planning procedure, in each pushing step, an action is selected by a greedy planner minimizing the proposed function defined in Eq. 6. The core idea of the proposed method is to increase the priority of the actions in efficient action set $\{a^*\}^m$ in successive pushing steps so as to reduce the possibility of sliding contact between the pusher and the object. Fig. 8 explains the idea of the cost function. Firstly, we find the action that are most likely translate the object to the target among $\{a^*\}^m$ using Eq. 7. $\mathbf{v}_p(a_i)$ represents the predicted object displacement vector for action a_i. ${}^t\mathbf{v}_d$ is the required object displacement vector calculated by subtracting the object's current position vector from its target position vector. ${}^ta^*$ is the action that maximizes the dot product between $\mathbf{v}_p(a_i)$ and ${}^t\mathbf{v}_d$. Meanwhile, we calculate the expected object displacement after executing the pushing action a_i, represented by ${}^{t+1}\mathbf{v}_{d|a_i}$ in Eq. 8. If the magnitude of ${}^{t+1}\mathbf{v}_{d|a_i}$ is 0, the object can be translated to the target position exactly by executing a_i. After that, we calculate the predicted object displacement vector for ${}^ta^*$ after executing a_i using Eq. 9. $\theta_p(a_i)$ in Eq. 9 represents the predicted object rotation for a_i. $\mathbf{R}(\cdot)$ returns a 2×2 rotation matrix. Finally, we compute the cost for each pushing action a_i in the representative action set $\{a_r\}^n$ using Eq. 6. The first term is the same as Eq. 8. The second term is the

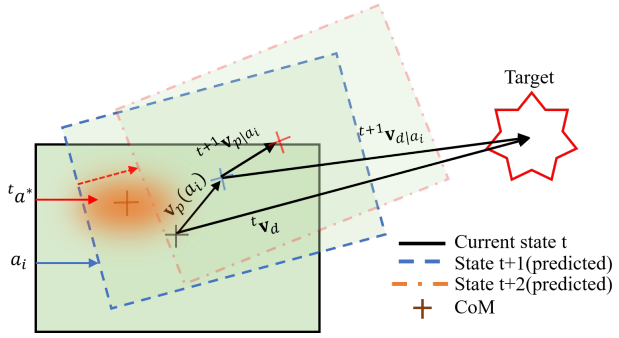

Fig. 8 Illustration of the proposed planning method. a_i is the pushing action applied to the object at the current time step t. $\mathbf{v}_p(a_i)$ and ${}^t\mathbf{v}_d$ are the predicted and expected object displacement, respectively, at t. ${}^ta^*$ is the action selected based on Eq. 7. ${}^{t+1}\mathbf{v}_{d|a_i}$ is the expected object displacement at $t+1$ time step, when applying a_i calculated by Eq. 8. ${}^{t+1}\mathbf{v}_{p|a_i}$ is the predicted object displacement, when applying ${}^ta^*$ at $t+1$ time step by Eq. 9.

λ weighted cosine similarity between ${}^{t+1}\mathbf{v}_{d|a_i}$ and ${}^{t+1}\mathbf{v}_{p|a_i}$. This term plays a core role in adjusting the object's orientation so as to increase the priority of the selected efficient action ${}^ta^*$. If λ is high, the planner will pay much attention to adjusting the object's orientation, which turns out to be inefficient. In this work, we normalize the first term by the mean magnitude of predicted object displacement and set λ to 1. The planning procedure will be terminated if the position error meets the requirement or the number of pushing steps exceed the requirement.

$$\phi(a_i) = \|{}^{t+1}\mathbf{v}_{d|a_i}\| - \lambda \frac{{}^{t+1}\mathbf{v}_{p|a_i} \cdot {}^{t+1}\mathbf{v}_{d|a_i}}{\|{}^{t+1}\mathbf{v}_{d|a_i}\| \|{}^{t+1}\mathbf{v}_{p|a_i}\|} \tag{6}$$

$${}^ta^* = \arg \max_{a_i \in \{a^*\}^m} \mathbf{v}_p(a_i) \cdot {}^t\mathbf{v}_d \tag{7}$$

$${}^{t+1}\mathbf{v}_{d|a_i} = {}^t\mathbf{v}_d - \mathbf{v}_p(a_i) \tag{8}$$

$${}^{t+1}\mathbf{v}_{p|a_i} = \mathbf{R}(\theta_p(a_i)) \cdot \mathbf{v}_p({}^ta^*) \tag{9}$$

5 EXPERIMENTS

5.1 Training Dataset

As a training step for the few-shot learning model, we create a dataset that consists of $\{M, priors, test, label\}$ tuples. For each tuple, M represents the object mask specifying the initial object pose. Object mask is obtained by cropping original mask image around the object center with $(100, 100)$ pixels, as it should fully include the biggest object in **SimPush**. The biggest object occupies around 90×90 pixels in the image frame. The pushing priors contain a series of pushing actions with known effects on the changes in object state, while the test contains actions and the label contains the outcome of test actions. In each tuple, all of the actions are applied to the same object with the same pose.

In **SimPush**, there are about 180 pushes for each object with specific physical property. For generating the training dataset, We randomly select 30 pushes to form a tuple from the 180 pushes. For each tuple, the orientation of object mask is randomly selected. Twelve of pushes are used as pushing priors and the remaining ones are the test actions and the corresponding labels. This procedure is repeated 50 times for each object. Because there are 200 different physical properties for each object shape, we obtain around 10K tuples for each object shape. We use 57 shapes for training and 12 shapes for testing. In total, the training set contains more than 570K tuples and the test set contains around 120K tuples.

5.2 Baseline and Ablation Models

5.2.1 Baseline models

We compare our model with the following baseline models.

- **NaiveCNN** takes all the features containing the object's mask image, action maps, a 8-D vector which consists of position in object mask, location of CoM, surface friction μ_s, contact friction μ_c, mass, and a scale ratio of pre-specified nominal inertia as input and outputs the ΔO. It consists of three sub-modules: The first one is a convolution network the same as the one in the push embedding module. The second one is a fully connected layer that has $8, 128, 256$ dimensions to process low-dimensional vector. At the end, we use another FCN with $512, 256, 128, 3$ to predict object motion.
- **Push-Net** takes historical pusher-object interactions into consideration to encode the transformation of the object state. We adapt it to explicitly predict object motion, taking 18 object masks and actions as input and outputting the action outcome for the current object state. We also add the loss term of the CoM to the loss function [34].

5.2.2 Model Ablations

We investigate contributions of object mask, action maps, and attention layers to the proposed model. We implemented the following three ablation models.

- **Model I (feature)** is used to find out how much the object mask contributes to the object motion prediction. We replace the push embedding modules with two 3-layer FCN modules of sizes (7, 128, 256) and (4, 128, 256), respectively. The input for embedding pushing priors are 7-D vector which consists of 4-D action vector and 3-D vector representing pose changes in object state. while the input to the embedding modules of the test only contains action vector. We keep other settings the same as the proposed model.
- **Model II (without action maps)** is used to show the importance of action maps. Actions are represented by a 4-D vector instead of action maps. In order to reuse pre-trained layers of ResNet50, we tile the object mask into 3 channels. Push embedding module takes the tiled object mask and action vector as input to output a 256-D vector. We keep other settings the same as the proposed model.
- **Model III (without residual attention module)** is used to show the contribution of residual attention modules of the decoder. We directly concatenate f_m, f_q, f_c together and feed them into MLP to predict object motion.

5.3 Training

We implemented all the models using PyTorch. We set the batch size to 128 for NaiveCNN and 32 for other models. The Adagrad optimizer [13] was used, where the learning rate was set to 0.001 with exponential time decay. For the baseline and ablation models, we used mean squared error as the loss function, since we only compare the performance on predicting pushing action effect. We stopped training at the $20th$ epoch for all models, as there were no significant changes in the loss curve thereafter. The training was conducted using an NVIDIA GTX 3090 GPU.

5.4 Real Experiments

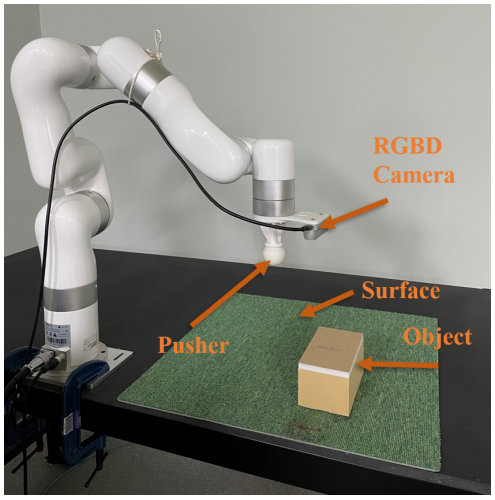

Fig. 9 Real experimental setting.

We evaluate the proposed object motion prediction model in an experimental setting using the XArm 5 Lite robot as shown in Fig. 9. The robot arm is equipped with the in-house built pusher and an RGBD camera (Intel RealSense D450) mounted at the distal end of the arm. We generate a

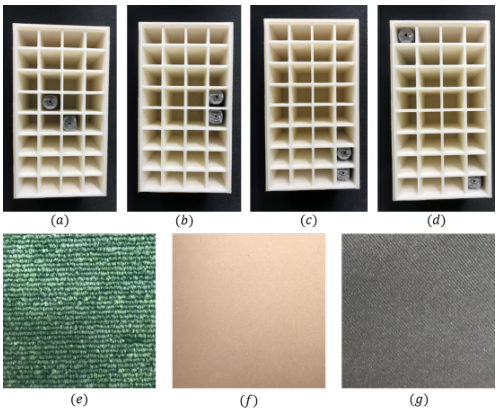

Fig. 10 Different lead block positions ((a)-(d)) and surface frictions ((e) carpet, (f) foam, and (g) cloth)

Fig. 11 Real novel objects used in our experiment.

point cloud from the camera and re-project it to the surface where the object is placed to get the object mask.

For the first experiment, we 3D printed a CoM controllable box of size $14 \times 7 \times 6cm$, with 4×8 grids inside the box as shown in Fig 10. We used 3 different surfaces made of artificial carpet, foam, and cloth having different friction coefficients and textures. We opted 4 different patterns of lead block positions in the box changing its CoM and pushed the box across the surfaces. The box has around 200 grams, and each lead block has approximately 80 grams. For each combination of CoM pattern and surface, the pusher executes a linear motion with $3cm$ to push the object 30 times at different contact points and pushing directions. Finally, we collect $30 \times 3 \times 4$ pushes for all the combinations of CoM settings and surfaces. We select 12 pushes as the pushing priors and the remaining ones as the test for each CoM setting and surface combination, using the model purely trained by the simulated data to predict the pose change of the pushed object. For the second experiment, we collected pushes for the three unknown objects shown in Fig. 11. They are complex in terms of shape and frictional contact with the floor surfaces. We push each object 30 times over the carpet surface, while keeping other parameters the same as the first experiment.

5.5 Planning

5.5.1 planning in simulation

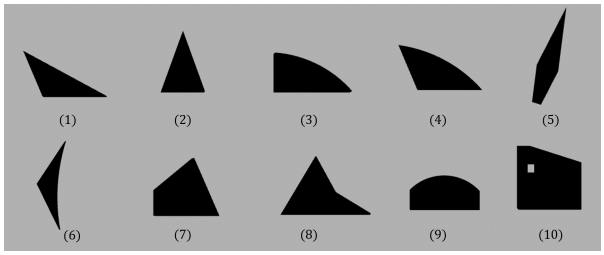

Fig. 12 Test objects used for planning simulation.

Firstly, we evaluate the proposed planning method in the simulation environment using 10 different objects shown in Fig. 12, each of which has 40 different physical properties including the CoM, inertia, weight, and contact friction. We conduct 20 push experiments for each object with specific properties that reach up to 8K push experiments in total. Each push experiment is defined by the distance between the initial and target positions of the object and the direction approaching the target. The initial position and orientation are kept the same and the distance is set to 300 millimeters, while the direction is uniformly sampled from $(-180^{\circ}, 180^{\circ})$. The pushing task is considered as success if the position error is less than the threshold within 14 pushing steps, otherwise as having failed. We choose 14 steps, as the mean translation distance per pushing is 22.2 millimeters when the pushing directions lie in the range of $(-60^{\circ}, 60^{\circ})$ *w.r.t.* the surface normal direction of the contact. The threshold was set to $3cm$ the same as the pushing length of an interval. The accuracy is the ratio of the total number of pushing tasks completed with a number of pushing steps lower than the threshold to the total number of conducted pushing tasks for each object.

Before the planning procedure, the robot interacts with the object a few times. Fig. 13 illustrates an example of the push affordance map generation procedure mentioned in Fig. 7. The robot pushes the object with unknown physical properties 12 times at different contact points with a direction sampled from $(-60^{\circ}, 60^{\circ})$ *w.r.t.* the surface normal direction. Each push length is $3cm$. Then, the applied pushing action and the resulting changes in object pose are used as the pushing priors to infer the resulting object motion and the prediction uncertainty for the representative actions $\{a_r\}^n$. As mentioned in 4.3, we used 5 different pushing directions *w.r.t.* the surface normal vector. We used 500 representative pushing actions in total, and the push affordances can be efficiently calculated on GPU in less than 1 second. Based on the push affordances and uncertainty predicted, and the top 100 representative action candidates, we

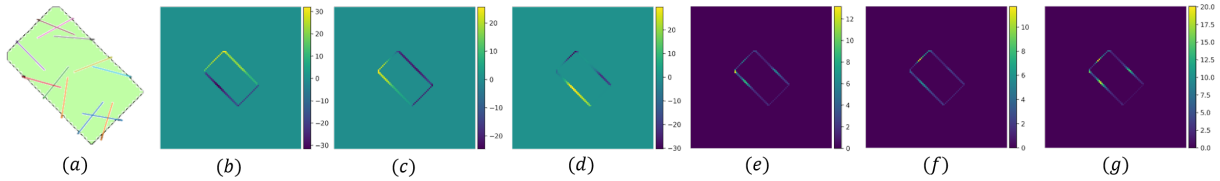

Fig. 13 Illustration of pushes: (a) encoding object dynamics, (b)-(g) generated push affordance map. Twelve pushes are conducted around the contour of the object along with a random direction within (-60, 60) *w.r.t.* the surface normal. (b) to (d) are the prediction object motion (translation and rotation) for the pushes along with normal direction to the outline. (e) to (f) are the predicted uncertainty *w.r.t.* the translation along x y axis as well as the rotation. In general, the predicted uncertainty for rotation tends to be larger than translation.

apply the non-maximum suppression algorithm to get a set of efficient actions represented by $\{a^*\}^m$.

In the planning procedure, the proposed cost function in Eq. 6 quantifies each representative action. We select the top 5 action candidates and use the softmax function to transform uncertainties associated with these candidates into a probability distribution. Finally, one action can be sampled from the distribution to be executed.

For the baseline method, prior pushing collection and push affordance map generation procedure were kept the same as the proposed method. However, the baseline method selects the top 5 ranked actions only minimizing the first term in Eq. 6. In other words, the baseline method only select actions that can minimize the translation error from the current to target positions.

5.5.2 planning on real platform

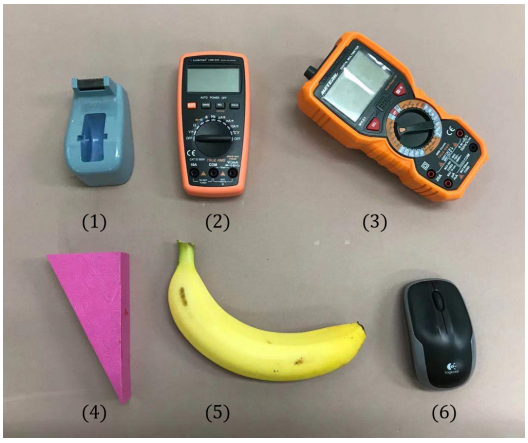

Fig. 14 Test objects used for planning on real experiment setting.

We evaluated the proposed method in real experimental settings on the foam surface with isotropic friction. We consider 6 different objects given in Fig. 14. Among them, the motions of *test-object-exp* 4 and *test-object-exp* 5 are difficult to predict, since *test-object-exp* 4 is pretty light (less than $5g$) and *test-object-exp* 5 has high contact friction across the foam. We conduct 16 pushing experiments for each object. The initial positions are identical, however the initial orientation is sampled uniformly in $(-180^{\circ}, 180^{\circ})$. We followed the same procedure as in simulation.

6 EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Model Prediction Result

The performance of the models implemented on the test set are given in Table 3. The last two columns represent prediction errors on translation and rotation. We use the $5th$ to $95th$ quantiles to represent the mean, standard deviation, maximum, and minimum. The average translation and rotation of the objects in **SimPush** are $19.13mm$ and $10.65°$, respectively.

The performance of two Push-Nets [34] are provided in Table 3. Here we only show the performance of the last time step. These two models perform less accurate than NaiveCNN. This is mainly due to the fact that NaiveCNN takes the internal parameters as input so that it also can learn how the internal parameters affect object motion. On the other hand, the result shows that the Push-Net trained with auxiliary CoM target performed similarly to the Push-Net without auxiliary learning in translation, yet worse in rotation. It can be conjectured that CoM plays a less dominant role in object motion. This is consistent with our observation that the motion of an object is slightly affected by the CoM when it has a much large object moment of inertia. Compared with the proposed model, all of the baseline models perform less accurately both in translation and rotation.

In the ablation study, we compare the proposed model to the following alternatives.

- **Model I (feature)** with other ablations; the models using object mask images perform better, possibly due to recent advances in CNN for push embedding. Therefore, the models have flexibility to encode the relation between pushing action and object shape.
- **Model II (no action maps)** with the proposed model; the action maps greatly helped improve the performance in predicting both translation and rotation.
- **Model III (no residual attention module)** with the proposed model; the cascaded residual attention model

Table 3 Prediction error on the test data of all the implemented models.

Models	translation (mm)				rotation (degree)			
	mean	std	max	min	mean	std	max	min
NaiveCNN	3.94	2.53	11.05	0.72	3.15	2.64	12.37	0.20
Push-Net (without auxiliary)	5.06	3.52	14.34	0.77	4.32	3.60	16.49	0.28
Push-Net	5.05	3.50	14.77	0.89	4.58	3.72	16.87	0.25
Model I (feature)	3.88	2.82	11.68	0.69	3.48	2.96	13.96	0.21
Model II (without action maps)	3.44	2.52	11.04	0.63	2.94	2.68	12.98	0.17
Model III (without residual attention module)	3.47	2.51	10.56	0.61	2.77	2.51	12.27	0.16
Proposed Model	**3.08**	**2.19**	**10.71**	**0.48**	**2.52**	**2.43**	**11.64**	**0.12**
Proposed Model(6 cm pushing)	**2.81**	2.31	11.01	0.39	3.97	4.06	19.61	0.17

Table 4 Translation prediction errors (in mm) for CoM-controllable Box

	Carpet				Foam				Cloth			
	mean	std	max	min	mean	std	max	min	mean	std	max	min
CoM1	3.1	1.3	6.4	1.72	3.15	1.78	6.89	0.78	4.98	2.32	9.96	1.71
CoM2	5.29	2.31	8.78	1.55	4.52	1.93	7.53	1.5	4.86	3.25	10.52	1.08
CoM3	3.61	1.29	6.49	1.78	3.85	1.84	6.55	1.21	5.56	3.5	13.19	1.84
CoM4	3.92	1.5	6.29	1.79	3.84	1.88	7.42	1.57	6.24	2.63	11.29	2.6

Table 5 Rotation prediction errors (in degree) for CoM-Controllable Box

	Carpet				Foam				Cloth			
	mean	std	max	min	mean	std	max	min	mean	std	max	min
CoM1	4.17	3.38	12.96	1.03	3.36	2.05	7.2	0.4	3.38	2.09	6.63	0.53
CoM2	3.99	2.49	9.57	1.0	3.28	1.94	6.55	0.61	4.46	3.0	11.55	0.47
CoM3	3.02	2.55	9.13	0.2	2.48	1.64	5.68	0.19	3.92	2.53	10.35	0.47
CoM4	2.95	1.63	6.26	0.84	2.89	2.44	8.1	0.61	3.29	2.31	7.83	0.41

Table 6 Prediction errors for objects motion in Fig. 11

	translation (mm)				rotation (degree)			
	mean	std	max	min	mean	std	max	min
obj1	4.23	1.95	8.88	1.30	4.03	4.08	13.53	0.20
obj2	3.34	1.72	6.30	0.57	2.92	2.23	7.71	0.05
obj3	4.67	2.36	9.28	0.26	3.72	3.07	11.84	0.59

selectively combines them and leads to more accurate pushing performance, instead of concatenating the input from a different source and feeding to MLP.

To verify if the proposed model can predict object motion for pushes with a larger magnitude, we collect a dataset that contains $6cm$ pushes using the object shown in Fig. 4 by following Algorithm 1. Then we re-scale the object mask and pushing lengths by a scale factor of 0.5 since the pushing magnitude is doubled. After that, we follow the same procedure as in Section. 5.1 to prepare the test set and evaluate the proposed affordance models to this test set. The result is shown in the last row of Table. 3. We observe that rotation prediction error becomes larger, however there is no difference in the translation prediction error. Table 4 and Table 5 show the result of predicting the motion of the CoM controllable box pushed over different surfaces. Because of the simulation-reality gap, noisy mask image, and calibration error, the results are less accurate than the one shown in Table 3. However, the proposed model purely trained by the simulation dataset predicted pretty close. The average translation and rotation of all the combinations are $20.8mm$ and $11.28°$, and the mean prediction errors on translation and rotation are $4.17mm$ and $3.43°$, respectively. The worst accuracy results were obtained both in translation and rotation when the object was sliding on the cloth surface, mainly due to cloth deformation.

We show the performance of our model on real novel objects in Table 6. The average translation and rotation of the objects are $23.13mm$, $21.41mm$, $19.72mm$, and $11.71°$, $11.28°$, $12.93°$, respectively. Similarly, compared with the result in the first experiment, there is no significant difference both in translation and rotation error.

6.2 Planning Evaluation

The comparative performance of the proposed planning and the baseline methods is presented in Table 7 with the mean, standard deviation and accuracy of pushing steps. Fig. 15 shows the accumulated movement of the objects. Ideally, the accumulated movement will be very close to $0.3m$, when the object approaches the target straight. Lower accumulated movement means less distance traveled on the plane. Compared with the baseline method, our method achieved lower accumulated movement on average for the objects shown in Fig. 12. From the results shown in Table 7, we found that the proposed planning yields a smaller number of pushes except for *test-object-sim* 9 in which both of the methods have rel-

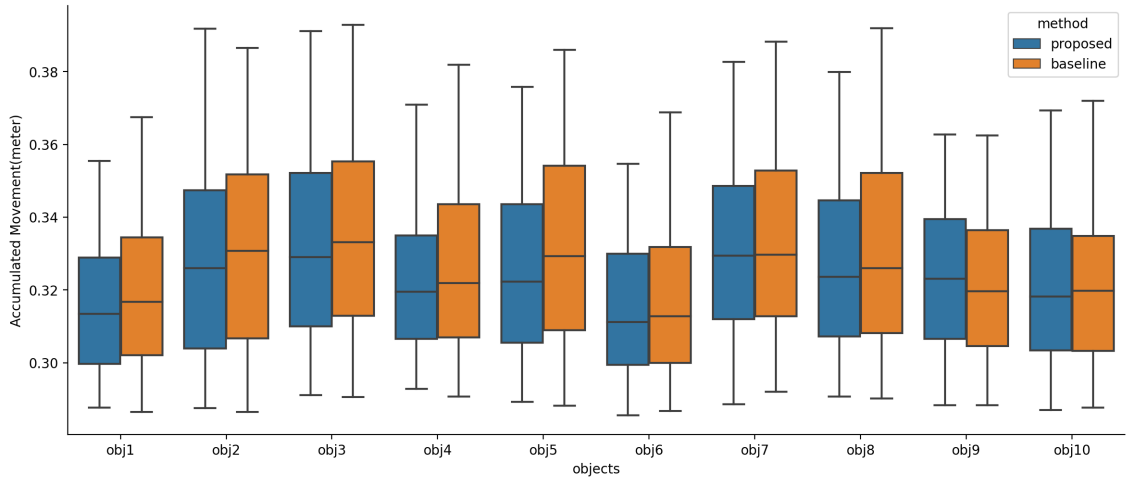

Fig. 15 Accumulated movement of objects shown in Fig. 12 for the planning task. The line inside each color box is the mean of the accumulated movement.

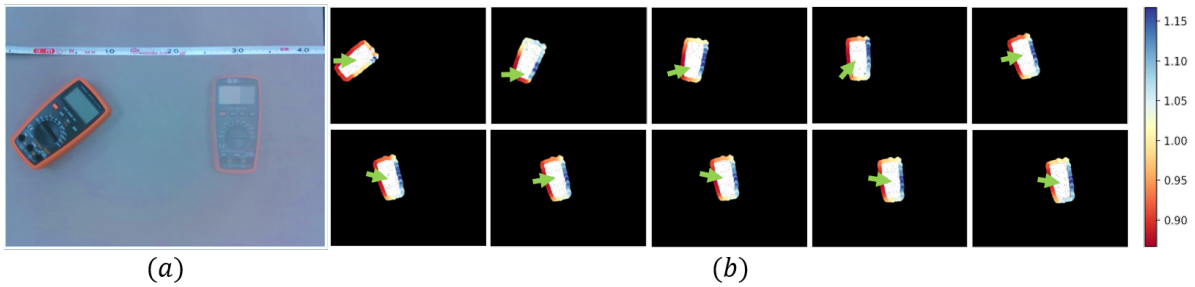

Fig. 16 An example of pushing *test-object-exp* 3 in Fig. 14 to a new position. (a) the initial and target poses $30cm$ away. The initial orientation is randomly selected. (b) pushing action denoted by an arrow and cost denoted as heatmap at each contact point in each step. There are 5 costs associated with each contact point as there are 5 representative pushing actions. Here only the minimum cost of each contact point is demonstrated.

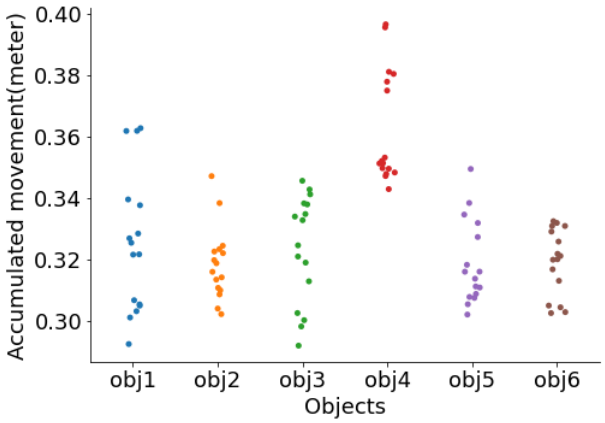

Fig. 17 Accumulated movement of objects shown in Fig. 14.

atively good performance. Moreover, the proposed planning shows a lower standard deviation, leading to the fact that the method is less sensitive to the object's initial orientation as well as different physical properties. The baseline method is sensitive to the object shape than the proposed method, as it suffered from low accuracy on *test-object-sim* 1 and *test-object-sim* 6.

We found that the baseline model performs obviously worse than the proposed method for narrow and long objects with a small number of edges such as *test-object-sim1* and *test-object-sim6*. Especially, in the case that the principal axis of this type of object is aligned with the desired translation direction, the baseline method tends to choose to push actions that cause sliding contact between the pusher and the object so that the object motion after each pushing is small. On the other hand, the proposed method performed robustly for this type of object, favoring actions to rotate the object in such a way to make the principal axis of the object perpendicular to the desired translation direction.

Fig. 17 shows the accumulated movement for the test objects on the real platform. It can be seen that the accumulated movement of *test-object-exp* 4 is the largest among the test objects. This is mainly due to the fact that *test-object-exp* 4 is too light resulting that the contact situation between object and foam surface introduces more randomness in object motion. We use the same metric for calculating the accuracy. We found that the accuracy in total is 94.79%. All the objects were successfully pushed to the target within 14 pushes except *test-object-exp* 4 and *test-object-exp* 5 for

Table 7 The mean, standard deviation of the pushing steps as well as the accuracy for objects shown in Fig. 12

Object	Method	Mean	Std	Accuracy (in percent)
obj1	proposed	**11.33**	**2.40**	**90.50**
	baseline	12.18	3.24	78.12
obj2	proposed	**10.77**	**1.68**	**97.25**
	baseline	10.79	1.79	96.50
obj3	proposed	**10.68**	**1.70**	**96.50**
	baseline	10.94	2.15	92.50
obj4	proposed	**10.90**	**1.78**	**95.62**
	baseline	11.33	2.24	90.75
obj5	proposed	**10.99**	**1.59**	**96.88**
	baseline	11.30	1.97	93.62
obj6	proposed	**11.60**	**2.62**	**87.12**
	baseline	12.22	3.21	76.75
obj7	proposed	**10.53**	**1.51**	**98.79**
	baseline	10.77	1.76	96.21
obj8	proposed	**11.07**	**1.78**	**96.00**
	baseline	11.39	2.00	92.38
obj9	proposed	10.83	1.70	97.29
	baseline	10.65	1.46	**100.00**
obj10	proposed	**11.09**	**1.64**	**97.43**
	baseline	11.30	1.72	95.14

which 3 and 2 pushing tasks are failed, respectively. *Test-object-exp* 5 tends to remain in its position during pushing because of the high contact friction. Notably, the proposed method still performed well for *test-object-exp* 3 even it has complex contact conditions with the surface.

Fig. 16 shows an example of pushing *test-object-exp* 3 in Fig 14. In this example, the robot pushed 10 times to translate the object to the target position. One interesting thing we found is that, at the steps from 1 to 4, the robot tried to adjust object orientation choosing several different contact points. After that, the robot focused on translating the object, while adjusting the object's orientation by slightly changing the pushing direction or contact point. Evidently, this is aligned with our original intention, as we aim to realize efficient pushing by changing the orientation of the object, which makes translation efficient.

7 CONCLUSION

Following the recent works in [5, 51], we presented a large-scale simulation dataset called **SimPush** containing a vast variety of objects diverse in shape and size. We simulated planar pushing under hundreds of varying conditions of contact friction, surface friction, mass, inertia, and CoM. Eventually, this dataset has more than 2 million push examples. Furthermore, we proposed a novel method to encode pushes, which greatly improved the model performance. Based on the encoder-decoder structure, we developed cascaded residual attention modules to combine features from different sources. Compared with the model that concatenates all features together and feeds into MLP, our model efficiently combined the features and helped the representation learning. Based on the proposed single-step prediction model, we proposed a novel planning method that can deal with objects with unknown physical properties.

We evaluated the proposed model purely trained by **SimPush** on a real platform. We designed a CoM controllable box pushed by a robot arm across different surfaces. Due to the noisy input and the simulation-to-reality gap, our model was not on a par with the results in simulation. However, our model predicted object motions with reasonable accuracy. We pushed three unknown real objects across the unknown frictional floor surface to challenge our model. Notably, the proposed model performed encouragingly well. Using the large-scale dataset, our model efficiently learned to make use of pushing priors to infer the novel action outcome. Compared with the model which depends on the quality of identification system, our model has proven robust in complicated object pushing. We evaluated the proposed planning method both in simulation and on a real platform. The planning method not only minimizes the position error, but also takes the efficiency of pushing into account. It should be emphasized that the proposed model performs as well in the unknown real world as in simulations with small samples of real world evidence.

In this paper, we studied the linear single-point contact pushing from the aspects of both predicting object motion and planning pushes to translate the object to the desired position. As a future direction of this research, a direct extension would be predicting push affordances when there are obstacles near the object. The current push planning method was assumed to use fixed length actions. Push planning with variable length actions will be an interesting problem for future research. For instance, pushing an object for a long distance initially toward a goal position, then pushing the object a short distance to adjust finely the object pose may lead to an efficient planning approach to reduce the number of pushes required. In addition, different types of contact (*e.g.*, multi-point contact) can also be an extension of this work. Such types of contact could improve pushing efficiency but also make prediction model challenging since the dynamics of contact tends to become more complicated.

References

1. Agrawal, P., Nair, A.V., Abbeel, P., Malik, J., Levine, S.: Learning to poke by poking: Experiential learning of intuitive physics. In: Advances in Neural Information Processing Systems, pp. 5074–5082 (2016)
2. Allevato, A., Pryor, M., Thomaz, A.: Multi-parameter real-world system identification using iterative residual tuning. In: Proceedings of the ASME International Design and Technical Conference. St. Louis, MO (2020)
3. Allevato, A., Short, E.S., Pryor, M., Thomaz, A.: Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In: L.P. Kaelbling, D. Kragic, K. Sugiura (eds.)

Proceedings of the Conference on Robot Learning, *Proceedings of Machine Learning Research*, vol. 100, pp. 445–455. PMLR (2020). URL http://proceedings.mlr.press/v100/allevato20a.html
4. Arruda, E., Mathew, M.J., Kopicki, M., Mistry, M., Azad, M., Wyatt, J.L.: Uncertainty averse pushing with model predictive path integral control. IEEE-RAS International Conference on Humanoid Robots pp. 497–502 (2017). DOI 10.1109/HUMANOIDS.2017.8246918
5. Bauza, M., Alet, F., Lin, Y.C., Lozano-Pérez, T., Kaelbling, L.P., Isola, P., Rodriguez, A.: Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgb-d video. arXiv preprint arXiv:1910.00618 (2019)
6. Byravan, A., Fox, D.: SE3-nets: Learning rigid body motion using deep neural networks. Proceedings - IEEE International Conference on Robotics and Automation (3), 173–180 (2017). DOI 10.1109/ICRA.2017.7989023
7. Chang, L., Smith, J., Fox, D.: Interactive singulation of objects from a pile. Proceedings - IEEE International Conference on Robotics and Automation pp. 3875–3882 (2012). DOI 10.1109/ICRA.2012.6224575
8. CM Labs Vortex Studio Academic. https://www.cm-labs.com/vortex-studio/software/vortex-studio-academic-access/. Accessed: 2020-09-30
9. Cosgun, A., Hermans, T., Emeli, V., Stilman, M.: Push planning for object placement on cluttered table surfaces. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4627–4632 (2011). DOI 10.1109/IROS.2011.6094737
10. Danielczuk, M., Mahler, J., Correa, C., Goldberg, K.: Linear push policies to increase grasp access for robot bin picking. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), pp. 1249–1256 (2018). DOI 10.1109/COASE.2018.8560406
11. Depierre, A., Dellandréa, E., Chen, L.: Jacquard: A large scale dataset for robotic grasp detection. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3511–3516 (2018). DOI 10.1109/IROS.2018.8593950
12. Dogar, M., Srinivasa, S.: A framework for push-grasping in clutter. Robotics: Science and systems VII **1** (2011)
13. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research **12**(7) (2011)
14. Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A., Levine, S.: Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. arXiv preprint arXiv:1812.00568 (2018)
15. Ebert, F., Finn, C., Lee, A.X., Levine, S.: Self-supervised visual planning with temporal skip connections. In: 1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings, *Proceedings of Machine Learning Research*, vol. 78, pp. 344–356. PMLR (2017). URL http://proceedings.mlr.press/v78/frederik-ebert17a.html
16. Ebert, F., Finn, C., Lee, A.X., Levine, S.: Self-supervised visual planning with temporal skip connections. In: CoRL, pp. 344–356 (2017)
17. Eitel, A., Hauff, N., Burgard, W.: Learning to Singulate Objects Using a Push Proposal Network. Springer Proceedings in Advanced Robotics **10**, 405–419 (2020). DOI 10.1007/978-3-030-28619-4_32
18. Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. arXiv preprint arXiv:1605.07157 (2016)
19. Finn, C., Levine, S.: Deep visual foresight for planning robot motion. Proceedings - IEEE International Conference on Robotics and Automation pp. 2786–2793 (2017). DOI 10.1109/ICRA.2017.7989324
20. Florence, P., Manuelli, L., Tedrake, R.: Self-supervised correspondence in visuomotor policy learning (2019)
21. Fragkiadaki, K., Agrawal, P., Levine, S., Malik, J.: Learning visual predictive models of physics for playing billiards. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings pp. 1–12 (2016)
22. Gao, Z., Elibol, A., Chong, N.Y.: A 2-stage framework for learning to push unknown objects. In: Joint IEEE International Conference on Development and Learning and Epigenetic Robotics, pp. 1–7 (2020). DOI 10.1109/ICDL-EpiRob48136.2020.9278075
23. Gao, Z., Elibol, A., Chong, N.Y.: Non-prehensile manipulation learning through self-supervision. In: IEEE International Conference on Robotic Computing, pp. 93–99 (2020)
24. Gao, Z., Elibol, A., Chong, N.Y.: Planar pushing of unknown objects using a large-scale simulation dataset and few-shot learning. In: 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), pp. 341–347 (2021). DOI 10.1109/CASE49439.2021.9551513
25. Goo, W., Niekum, S.: Local nonparametric meta-learning. arXiv preprint arXiv:2002.03272 (2020)
26. Goyal, S., Ruina, A., Papadopoulos, J.: Planar sliding with dry friction part 1. limit surface and moment function. Wear **143**(2), 307–330 (1991)
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
28. Hermans, T., Li, F., Rehg, J.M., Bobick, A.F.: Learning contact locations for pushing and orienting unknown objects. In: IEEE-RAS International Conference on Humanoid Robots, pp. 435–442 (2013)
29. Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., Teh, Y.W.: Attentive neural processes. arXiv preprint arXiv:1901.05761 (2019)
30. Kloss, A., Bauza, M., Wu, J., Tenenbaum, J.B., Rodriguez, A., Bohg, J.: Accurate vision-based manipulation through contact reasoning. In: IEEE International Conference on Robotics and Automation, pp. 6738–6744 (2020)
31. Kloss, A., Schaal, S., Bohg, J.: Combining learned and analytical models for predicting action effects from sensory data. The International Journal of Robotics Research p. 0278364920954896 (2018)
32. Kumar, K.N., Essa, I., Ha, S., Liu, C.K.: Estimating mass distribution of articulated objects using non-prehensile manipulation. arXiv preprint arXiv:1907.03964 (2019)
33. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research **37**(4-5), 421–436 (2018). DOI 10.1177/0278364917710318. URL https://doi.org/10.1177/0278364917710318
34. Li, J.K., Lee, W.S., Hsu, D.: Push-net: Deep planar pushing for objects with unknown physical properties. In: H. Kress-Gazit, S.S. Srinivasa, T. Howard, N. Atanasov (eds.) Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018 (2018). DOI 10.15607/RSS.2018.XIV.024. URL http://www.roboticsproceedings.org/rss14/p24.html
35. Lin, C., Grner, M., Ruppel, P., Liang, H., Hendrich, N., Zhang, J.: Self-adapting recurrent models for object pushing from learning in simulation. IEEE International Conference on Intelligent Robots and Systems pp. 5304–5310 (2020). DOI 10.1109/IROS45743.2020.9341076
36. Lynch, K.M., Maekawa, H., Tanie, K.: Manipulation and active sensing by pushing using tactile feedback. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 416–421 (1992)

37. Mason, M.T.: Mechanics and Planning of Manipulator Pushing Operations. International Journal of Robotics Research **5**(3), 53–71 (1986). DOI 10.1177/027836498600500303
38. Mavrakis, N., Ghalamzan E., A.M., Stolkin, R.: Estimating an object's inertial parameters by robotic pushing: A data-driven approach. IEEE International Conference on Intelligent Robots and Systems pp. 9537–9544 (2020). DOI 10.1109/IROS45743.2020.9341112
39. Pinto, L., Gupta, A.: Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. Proceedings - IEEE International Conference on Robotics and Automation **2016-June**, 3406–3413 (2016). DOI 10.1109/ICRA.2016.7487517
40. Rohmer, E., Singh, S.P.N., Freese, M.: Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2013)
41. Song, C., Boularias, A.: A probabilistic model for planar sliding of objects with unknown material properties: Identification and robust planning. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5311–5318 (2020). DOI 10.1109/IROS45743.2020.9341468
42. Stüber, J., Zito, C., Stolkin, R.: Let's push things forward: A survey on robot pushing. Frontiers in Robotics and AI **7**, 8 (2020)
43. Walker, J., Doersch, C., Gupta, A., Hebert, M.: An uncertain future: Forecasting from static images using variational autoencoders. In: European Conference on Computer Vision, pp. 835–851. Springer (2016)
44. Wang, C., Wang, S., Romero, B., Veiga, F., Adelson, E.: SwingBot: Learning physical features from in-hand tactile exploration for dynamic swing-up manipulation. IEEE International Conference on Intelligent Robots and Systems pp. 5633–5640 (2020). DOI 10.1109/IROS45743.2020.9341006
45. Wang, J., Hu, C., Wang, Y., Zhu, Y.: Dynamics learning with object-centric interaction networks for robot manipulation. IEEE Access **9**, 68277–68288 (2021). DOI 10.1109/ACCESS.2021.3077117
46. Wu, J., Lim, J.J., Zhang, H., Tenenbaum, J.B., Freeman, W.T.: Physics 101: Learning physical object properties from unlabeled videos. British Machine Vision Conference 2016, BMVC 2016 **2016-September**, 39.1–39.12 (2016). DOI 10.5244/C.30.39
47. Xu, Z., He, Z., Wu, J., Song, S.: Learning 3d dynamic scene representations for robot manipulation. CoRR **abs/2011.01968** (2020). URL `https://arxiv.org/abs/2011.01968`
48. Xu, Z., Wu, J., Zeng, A., Tenenbaum, J.B., Song, S.: Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. In: Robotics: Science and Systems (RSS) (2019). URL `http://www.zhenjiaxu.com/DensePhysNet/`
49. Xu, Z., Yu, W., Herzog, A., Lu, W., Fu, C., Tomizuka, M., Bai, Y., Liu, C.K., Ho, D.: Cocoi: Contact-aware online context inference for generalizable non-planar pushing. arXiv preprint arXiv:2011.11270 (2020)
50. Ye, Y., Gandhi, D., Gupta, A., Tulsiani, S.: Object-centric forward modeling for model predictive control. arXiv (CoRL), 1–13 (2019)
51. Yu, K.T., Bauza, M., Fazeli, N., Rodriguez, A.: More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 30–37 (2016)
52. Yu, W., Tan, J., Liu, C.K., Turk, G.: Preparing for the unknown: Learning a universal policy with online system identification. Robotics: Science and Systems **13** (2017). DOI 10.15607/rss.2017.xiii.048
53. Zeng, A., Florence, P., Tompson, J., Welker, S., Chien, J., Attarian, M., Armstrong, T., Krasin, I., Duong, D., Sindhwani, V., Lee, J.: Transporter networks: Rearranging the visual world for robotic manipulation (2021)