Doctoral Dissertation

A FEW-SHOT LEARNING APPROACH TO CONTROLLED PLANAR PUSHING OF NOVEL OBJECTS ON UNKNOWN FRICTIONAL SURFACES

Gao Ziyan

Supervisor Nak Young Chong

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

Septemper 2022

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. The prediction model was 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. Secondly, We exploit the few-shot learning model on object center of mass estimation as well as push planning for re-arranging object positions. For center of mass estimation problem, we estimate the center of mass (CoM) of an object by narrowing down its probable location with the proposed model and Mason's voting theorem. The result shows that the CoM estimation method has good mean squared error properties and small standard deviation. For push planning, 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 experimental results showed that the push planning method effectively reduces the number of pushes required to move unknown real objects to target positions. Apart from exploiting the few-shot learning model, we proposed to use dynamic pushing to estimate the center of mass online efficiently. We

conducted simulation experiment and the estimation process was conduced both in isotropic and an-isotropic frictional settings. Comparing with the result of quasi-static pushing for CoM estimation, dynamic pushing significantly outperformed both in isotropic and an-isotropic frictional setting. In addition, we propose the Zero Moment Two Edge Pushing (ZMTEP) method to translate a novel object without rotation to a goal pose. The proposed method enables a pusher to select the most suitable two-edge-contact configuration for a given object using the estimated CoM and the geometrical shape of the object. Notably, neither the friction between the object and its support plane nor the friction between the object and the pusher are assumed to be known. We evaluate the proposed CoM estimation and ZMTEP methods through a series of experiments in both simulation and real robotic pusher settings. The results show that the ZMTEP method significantly outperforms competitive baseline methods. **Keywords:** Planar Pushing, Object Center of Mass Estimation, Path Planning, Non-prehensile Manipulation, Data-driven Automation

Acknowledgment

First of all, I would like to express my sincere gratitude and respect to Prof. Chong who is my principal supervisor. I cannot approach this stage without his great help. He gave me endless encouragement and wise guidance to make the way of pursuing this degree enjoyable. At the beginning, he gave me high freedom to choose the research topic, then, he gave me right direction to start my research and offered everything that I need to conduct my research and experiment. His encouragement raised me up to be brave and spirited to difficulties encountered.

Then, I would like to appreciate my second advisor Prof. Elibol. In research, he empty his purse to help me how to conduct research, how to think like a researcher. His ”painful” questions shaped my thinking. In daily life, when I feel depressed, he always be there to encourage me.

Also, I would like to express my sincere gratitude to my parents. They spare their all to support me. Due to Covid-19, I haven't chance to come back to my hometown, I miss them very much. As an engineer, my father often discusses with me on the methodology of problem solving task, which gives me many insights when I was got stuck in research. My mother gives me endless love which I cannot repay. Her biggest wish to me is not to be someone who is rich or sciential but to be someone who has a joyous life.

I would like to appreciate my girlfriend Miss.Hanzi Lu. She accompanies me to experience joys and sorrows of the life and research. I cannot imagine the shape and color of life without her. It is her who always believes me no matter what challenges I was facing in research. In daily life, we experienced the beauty of Tirihama shore at dusk, the grande of Kamikouti, The cozy of bonfire, and the

tasty of three meals a day. All these experiences make me believe that she is the one whom I want to marry.

Last but not least, I want to thank Mr.HUANG Hsuan Tsung, he helped me greatly to design the hardware for conducting the experiment. In addition, I want to than Dr.Zhihao Shen and Mr.Yanhao Wang, they helped me for setting the real platform.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Moving an object from one location to another is a common task encountered in our daily life and industrial production lines. This task can be accomplished through pick-and-place, pushing, throwing, and similar others. Among these, pushing is an undemanding manipulation primitive [1] to locate and orient an object to the desired pose. Pushing neither needs form or force closure nor requires conforming to the object's geometrical shapes. Pushing is preferable to re-positioning an object not directly graspable within the given pose. Therefore, pushing has been used to complement other tasks such as object placement [2], object singulation [3, 4], and grasping [5–8].

Even though pushing has many attractive features, it exhibits multiple contact modes (sticking, sliding, and separation), indeterminacy due to unknown pressure distribution of the pushed object, and under-actuation. All of these introduce challenges in modeling, controlling, and planning.

Modeling for the pusher-slider system is important to facilitate efficient control and planning. For modeling the pusher-slider interactions, earlier works were focusing on employing analytic models to characterize the object behaviors given the external pushing force. For instance, Mason [9] proposed voting theorem to predict the object rotation direction without the knowledge of pressure distribution. Goyal *et al.* [10, 11] introduced the concept of limit surface to describe the relationship between the applied wrench and the object twist by assuming

minimum energy dissipation (MED). However, analytic models under-fits the real pushing interactions and many assumptions are made such as the known friction, perfect uniform ground, MED, Coulomb's friction law, *etc,*.

Recently, Stuber *et al.* [12] reviewed the methods for modeling the pusher-slider system with machine learning-based approaches that benefit from their enhanced performance in 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 [13, 14]. However, the datasets contain a limited number of objects that are most 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 [14].

There are studies showing that the knowledge learned from simulation can be transferred to real settings with (or without) a small amount of real data [15–19]. Depierre *et al*. [20] and Byravan *et al.* [19] 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 effect prediction problem. The **SimPush** dataset contains 69 objects (depicted in Fig. 3.1) with diverse dimensions and shapes that appear either convex or concave. We adopt quasi-static pushing to interact with the object.

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.

Even though the data-driven model can better model the pushing interactions, it is still challenging to deal with novel objects because of unknown physical properties, such as mass, the center of mass (CoM), friction, *etc,*. On the other hand, humans can quickly identify the object's behavior after a few interactions with that object. We wonder if it is possible to endow the robot with this capability to have a better generalization for modeling the pushing interactions for novel objects.

Based upon the recent development of Few-shot learning models [21] which can rapidly generalize to new tasks using few labeled samples, we adopt a few-shot learning model to predict push effects. The few-shot learning model leverages a small set of pushing priors, which can be collected in a self-supervised manner, aiming to infer the pushing effect for other pushing actions. This model can help deal with an infinite number of real-world objects using a limited dataset. In addition, 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.

If the pushing effect prediction model is available, how can we use it? In this work, we used the prediction model in two aspects: single-contact push planning and object CoM estimation. In this work, single-contact push planning deal with the problem of translating the novel object to its desired position. Specifically, we use the trained push effect model to predict the push effects 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 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 the 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.

For the object CoM estimation problem, we combine the prediction model with Mason's voting theorem to estimate the object CoM. Notably, neither the friction between the object and its support plane nor the friction between the object and the pusher are assumed to be known. Please note that Mason's voting theorem cannot be applied directly to this problem because of the absent knowledge of the friction cone at the contact position between the robot and the novel object. In this work, we examine the cases in which the probable region of object CoM can be uniquely specified by the pushing actions and the resultant rotation directions. Specifically, the probable region is initialized by the convex hull of the object shape, the proposed method uses a set of sampled pushing actions with predicted resultant object rotation directions to iteratively narrow down the probable region

of object CoM location. We evaluate the proposed CoM estimation through a series of experiments in both simulation and real robotic pusher settings. The result shows that the CoM estimation method has good mean squared error properties and small standard deviation.

We employ a prediction model and quasi-static pushing to deal with CoM estimation. However, there is considerable empirical evidence that even though the prediction model has decent performance in predicting the resultant object rotations of pushing action, it has less in-accurate performance for pushing actions which results in small amount of rotations. In addition, the quasi-static pushing may lead to inaccurate estimation due to the sliding contact because of the sensory noise, *i.e.,* in-accurate contact normal estimation due to the sensor resolution. On the other hand, dynamic pushing, which pushes the object at high speed, can shorten the time of sliding contact by introducing inertial force which can keep the object moving after terminating the movement of the pusher. One of the assumptions of Mason's voting theorem is quasi-static dynamics, we wonder if Mason's voting theorem still holds in dynamic pushing. In addition, instead of using a prediction model, we estimate the object's CoM online and minimize the number of pushes executed. We conducted a simulation experiment on twenty different objects and in different frictional settings (isotropic and anisotropic). The result showed that dynamic pushing significantly outperformed quasi-static pushing on CoM estimation task.

In the previous study, we directly leveraged the prediction model and heuristics to re-arrange the object by using single-contact pushing. However, if the object's CoM is available and more than a single contact configuration can be achieved, how can we exploit it to increase the efficiency of the object re-arrangement task? The study [22] done by Lynch and Mason claimed that the pusher-slider system becomes stable by introducing extra contacts. Specifically, it

shows that there is a set of pushing directions along which the slider's pose relative to the pusher remained fixed given known physical properties. They proposed to use line pushing with multiple contacts on the same edge of a polygonal slider and stabilized its motion [22–24].

Supposing that a robot pushes an object on a flat surface, and there is nearly zero friction between the robot and the object, line pushing becomes extremely difficult as there is little friction that can be exploited. It is therefore advantageous to use multi-edge contact whose normal force direction positively spans the desired pushing direction. A robot equipped with an adjustable stroke gripper can make multiple contacts that are not necessarily on the object's same edge. It should be also noted that when a pusher maintains line (or point) contact with a slider to move it around, non-holonomic constraints need to be addressed. Zhou *et al.* [24] showed that Dubin's curve is the time optimal trajectory under the assumption of sticking contact.

Instead of solving the trajectory planning problem in a complex way, we pose a new question: how can we select two contact points from two polygonal edges of an object with estimated CoM to slide it along a straight line to a goal pose? As the shortest distance between two points is a straight line, a solution to this problem yields a reasonably close optimal shortest path. To deal with this problem of double-contact selection, we analyzed the force and moment conditions of pure translation, and the moment labeling representation of the contact force, we propose a double-contact pusher-slider interaction, called Zero Moment Two Edge Pushing (ZMTEP), to translate a novel object to a goal pose.

In real-world pushing, measurement noise and CoM estimation error are unavoidable, leading to an incorrect contact configuration. Therefore, we discuss the tolerance of ZMTEP to noisy observation and error-prone CoM estimates. We found that the tolerance to the CoM estimation error increases, as the distance

between two contact positions increases. The tolerance to noisy observations increases up to a certain point and decreases as the contact point moves toward the endpoint of the object edge. A series of experiments are conducted to verify if the proposed CoM estimation and ZMTEP methods are valid and competitive compared to the two baseline methods in simulation and real settings. The results revealed that the CoM estimation error was low overall that can be well tolerated by the two-edge-contact configuration selected by ZMTEP.

The following assumptions are made throughout this paper:

- Both the robot's end effector (pusher) and the object (slider) are rigid, and interact quasi-statically.
- Object geometry can be extracted by a vision sensor.
- The object contact normal can be extracted.
- The plane in which the object lies is flat.
- Coulomb's friction law applies.

To summarize, the main contributions of this work are threefold:

- A large-scale and diverse planar pushing dataset that contains convex and concave objects with different physical properties.
- A push effect prediction model and a compact state representation for planar pushing.
- A push planning method integrating a push effect prediction model to translate the novel object to the desired position efficiently.
- An integrated CoM estimation method combining the motion prediction model and the voting theorem.
- A efficient CoM estimation method by dynamic pushing operation.
- The most suitable two-edge-contact pushing configuration for translating a novel object without rotation.

- An analysis of the tolerance to measurement noise and error-prone CoM estimates.

This paper is organized as follows.

- Chapter. 2, The research background related to pushing dataset generation, pushing interaction modeling, pushing planning, object inertial parameter estimation, as well as contact location selection for planar pushing, are represented.
- Chaper. 3 represents our simulation planar pushing dataset - **SimPush**. Firstly, the simulator and physical engine we used are introduced, then is the procedure of self-supervised dataset generation.
- Chapter. 4 shows the proposed pushing effect prediction model and the pushing primitive representation. The dataset generation based on **SimPush** for training the push effect prediction model is described. Then, we evaluate the prediction model both in simulation and in a real setting.
- Chapter. 5 introduces the several object CoM estimation methods. Firstly, the method combining the prediction model and Mason's voting theorem for CoM estimation is presented. Then is the CoM estimation method by using the predicted accumulated object rotation. After that, we present CoM estimation by using dynamic pushing. Finally, we evaluated all of the mentioned methods in simulation and evaluate the first method in the real setting.
- Chapter. 6 introduces the proposed push planning methods by exploiting the prediction model.
- Chapter. 7 presents double-contact configuration selection method for purely translating novel object. In this chapter, we analyze the sensory tolerance and CoM estimation tolerance capability of the contact configuration.
- Chapter. 8 summarizes the dissertation and draws several potential future

works.

Chapter 2

Literature Review

2.1 Dataset

Learning-based methods are capable of dealing with unseen objects using various datasets [14, 25–28] for training. Yu *et al.* [13] 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 [14] 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.* [29] proposed a pushing dataset that contains 57 thousands pushes to directly model the pixel motion in the image frame. Eitel *et al.* [30] proposed a pushing dataset for object singulation task. Pulkit *et al.* [31] 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 Push Interaction Modeling

Mason [9] analyzed the mechanics of quasi-static pushing and presented the voting theorem for determining the sense of rotation of a pushed object. Goyal *et al*. [32] introduced the limit surface reasoning the frictional forces with object motion. Kloss *et al.* [33] combined a deep learning model and an ellipsoid limit surface to improve both the generalization capability and accuracy. Zhou *et al.* [24] present a physics-based data-driven model to approximate the limit surface. Bauza *et al*. [14] and Goo *et al*. [34] used Attentive Neural Process (ANP) [35] to learn the object dynamics by incorporating pushing priors. Li *et al*. [18] 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 explicitly investigated. Fragkiadaki *et al.* [36] predicted ball motion under a pushing force by leveraging the most recent glimpse of the object-centered image patches.

Different studies have been conducted to predict object motion incorporating system identification methods [15, 37–41]. The object physical properties are usually either explicitly estimated [15, 39] or implicitly represented by neural networks [36, 38, 40, 41]. Specifically, Wu *et al.* [39] fed explicitly estimated physical parameters as input to an analytical model of physical system to estimate object motion. Wang *et al.* [38] proposed to collect the tactile feedback data when executing predefined motion to implicitly encode physical properties of novel objects.

In addition, studies [29, 42–44] put predicting action effects into the video

prediction scenario with the self-supervised learning method.

2.3 Push 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 [31, 45–47]. Specifically, Zeng *et al*. [45] 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*. [48] and Arruda*et al*. [49] employ the model predictive path integral approach to optimize the pushing action. The cross-entropy method is also intensively used in [3, 50, 50, 51]. These methods need roll-out simulation from current state which is computationally expensive. Cosgun *et al*. [2] 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*. [2] and Dogar *et al*. [7] 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*. [52] 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. Lin *et al.* [48] integrated a recurrent neural network for pushing interaction modeling into a model predictive control

framework.

2.4 Object Inertial Parameter Estimation

An object's inertial parameters can be identified by various techniques classified into three types: purely visual, exploratory, and fixed-object [53]. The fixed-object type requires a fixed connection between the robot and an object such as grasping, which is beyond the scope of this research.

The purely visual type directly estimates the inertial property using a vision sensor. Trevor *et al.* [54] proposed a deep learning model trained on a labeled dataset to estimate the volume and density of the object using a RGB-D image. Although this model has competitive performance on image-to-mass tasks compared with humans, the estimation result is highly affected by occlusion or light conditions.

The exploratory type requires the robot to interact with the object to measure the applied forces and object motion. Then, the inertial parameters are estimated by solving physics laws of motion. Yu *et al.* [55] used a two-finger pushing in a trial-and-error approach making the line of CoM pass between the fingers. This method may be inefficient when dealing with an arbitrarily shaped object. Mavrakis *et al.* [56] estimated inertial properties using a single-contact pushing and data-driven learning model. Kloss *et al.* [15] used an Extended Kalman Filter (EKF) to iteratively estimate object physical properties (such as the CoM, friction, and mass) based on the information extracted from the object mask. The EKF was implemented assuming an ellipsoid model [57] between the applied force and the resultant object motion. Song *et al.* [37] proposed to learn the coupled mass-friction parameters through minimizing the simulation-reality gap. This method required a set of hypothesized mass and friction models

and the object was coarsely approximated by rigidly-connected 2D small grids. Allevato *et al.* [58, 59] used a neural network to tune the inertial parameters of the physics engine based on the difference in observation from the real object motion. However, it was limited to known objects. Instead of relying on an approximation model or physical simulator, data-driven models were used to learn the causality between the object motion and the inertial parameters in some studies. Li *et al.* [18] used recurrent neural networks to estimate the CoM of the object. Xu *et al.* [40] proposed a disentangled learning module to implicitly encode an object's physical properties through robot-object interaction. Kumar *et al.* [17] employed a policy network to interact with an articulated object, and a predictor network to predict the mass distribution of the object.

In this work, we propose an exploratory type CoM estimation method for an arbitrarily shaped object using a robot equipped with a vision sensor. In contrast to other exploratory type methods, only a few robot-object pushes are needed with no *a priori* assumption about friction.

2.5 Contact Location Selection for planar pushing

Without focusing on the pusher-slider interaction models, some studies used data-driven methods to learn from historical pushing interactions [60–62]. Hermans *et al.* [60] presented a kernel's method to learn the contact locations for pushing an unknown object. However, only the shape features were considered with a state feedback controller. In contrast, our ZMTEP method selects contact locations considering both the geometrical shape as well as the CoM of an object. Li and Payandeh [63] proposed a parametric formulation to find a two-point contact configuration of equilibrium push in which relative rotation between the pusher and the object will not happen employing a simplified model of a known

object. Most notably, based on fewer assumptions, our method determines the most suitable two-edge-contact configuration, analyzing the mechanics of pure translation that exhibits high tolerance toward measurement and CoM estimation noises.

On the other hand, instead of relying on push interaction models, Lloyd and Lepora [64] employed tactile and proprioceptive feedback to push an object across planar and curved surfaces. Danielczuk *et al.* [6] presented and compared several pushing policies for object singulation in cluttered bin environments which did not consider pushing interactions.

Chapter 3

Large-Scale Planar Pushing Simulation Dataset

In this chapter, we present a large-scale planar pushing simulation dataset called SimPush shown in Fig. 3.1. Firstly, we detailed the procedure of the data collection by using a simulator. Then we compare SimPush with other available dataset.

3.1 Simulation Environment

The simulation environment was created with CoppeliaSim [65] as shown in Fig. 3.2. 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

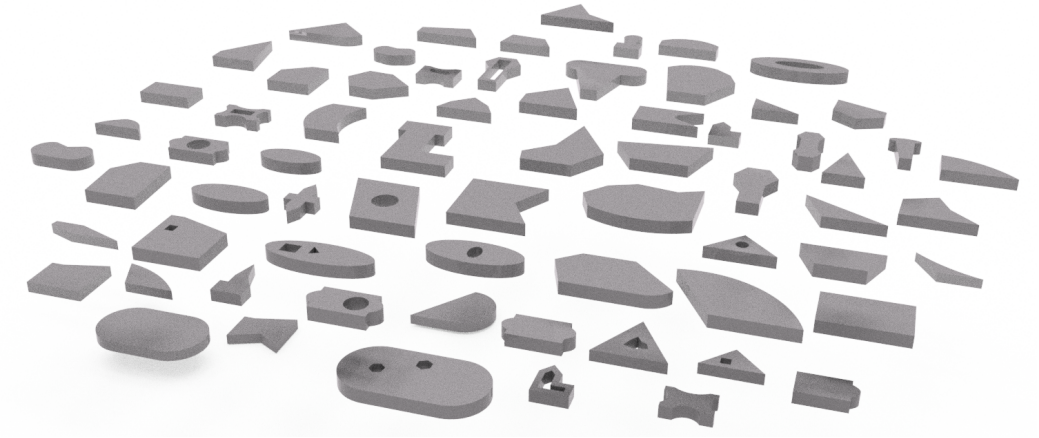


Figure 3.1: SimPush: Large-Scale Planar Pushing Dataset available at https://github.com//SimPush

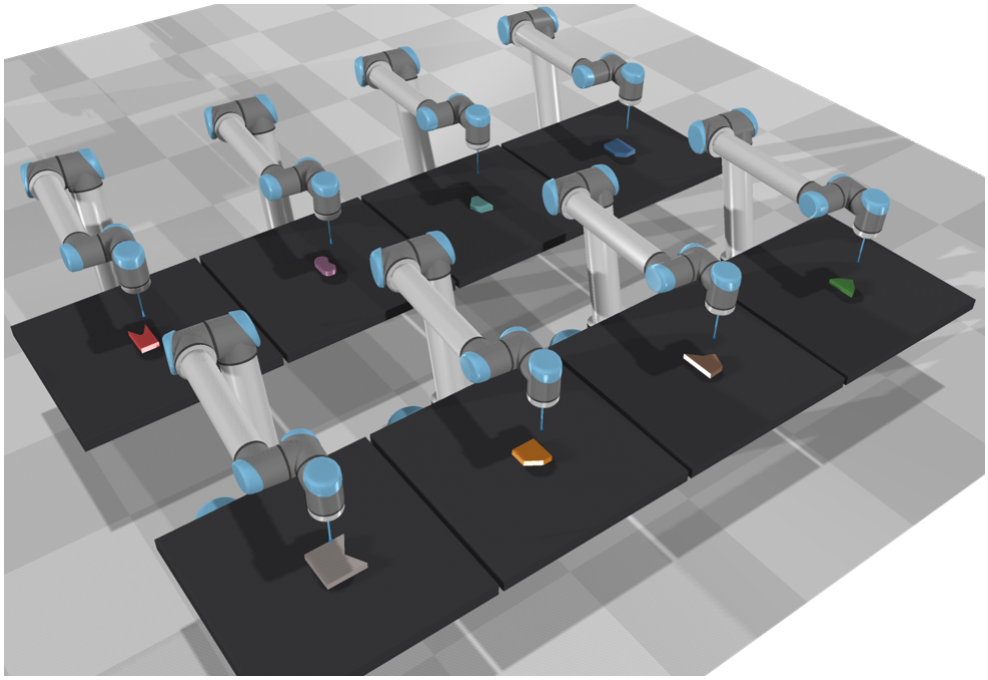


Figure 3.2: Planar pushing simulation environment developed in CoppeliaSim.

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 [66] 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_p), 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 3.2. 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.

In Table 3.1, **SimPush** is compared with the existing datasets.

Table 3.1: Comparison with existing push datasets

Dataset	**objects**	**surfaces**	**pushes**	**Platform**	**Size**
SimPush	**2760**	5	180	Simulation	**∼2M**
Omnipush [14]	250	1	250	Real	∼63K
Yu [13]	11	4	6000	Real	∼264K

Table 3.2: 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_p)	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

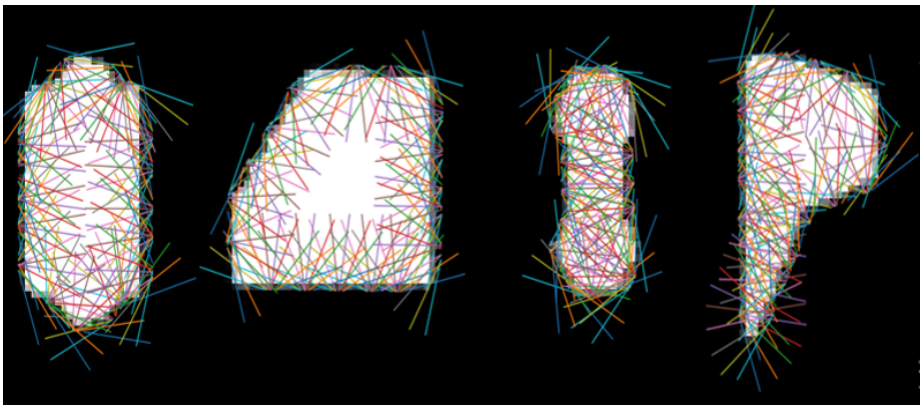


Figure 3.3: Pushing samples for different shapes. For each shape, 17-18 contact points are uniformly sampled across the object perimeter
, each of which has 10 different push directions. The line with different color represents different pushing direction *w.r.t.* the contact normal direction.

3.3 Dataset Collection Procedure

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. 3.3 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.

Algorithm 1: Data Collection Scheme

Input: object, $\mu_s, \mathbf{P}_{CoM}, \mu_p, I, m$

/* μ_s is the friction of the ground, $\mathbf{P}_{CoM}$ refers to the position of center of mass in object frame. μ_p is the friction of the pusher. I is the inertial tensor. m is the object mass. */

Output: cache

/* cache contains a series of applied pushing actions as well as the changes in object pose before and after the execution of the pushing actions. */

Set surface friction μ_s

Load object to the ground, determine initial object pose O_{init}, set inertial parameters $\mathbf{P}_{CoM}$, μ_p, inertia I, and mass m.

Sample k points $\{\mathbf{cp}_i\}_{i=1,...k}$ and $\{\mathbf{n}_i\}_{i=1,...k}$ uniformly across the object perimeter.

Sample n push directions $\{\theta_{p_j}\}_{j=1,...,n}$ in range of $[-\frac{3\pi}{4}, \frac{3\pi}{4}]$ *w.r.t.* the normal of each contact point $\{\mathbf{cp}_i\}_{i=1,...k}$.

cache$\leftarrow \emptyset$

cache.append(O_{init})

for $i = 1, ..., k$ **do**

  for $j = 1, ..., n$ **do**

    Reset object to $\mathbf{O}_{init}$.

    $\varepsilon \leftarrow$ Random-Sample()/* small perturbation */

    Move pusher to $\mathbf{cp}_i + \varepsilon$.

    Push object $3cm$ along the direction specified by θ_{p_j} and $\mathbf{n}_i$ w.r.t the contact normal.

    Get object state $\mathbf{O}_{ij}$

    cache.append($\mathbf{cp}_i + \varepsilon, \mathbf{n}_i, \theta_{p_j}, \mathbf{O}_{ij}$)

Chapter 4

A few-shot Learning Model for Pushing Effect Prediction

We formulate the push effect prediction problem as follows: given m pushing priors $\{\mathbf{cp}_i, \mathbf{n}_i, \theta_{p_i}, \Delta O_i\}_{i=1,\ldots,m}$ and n test data $\{\mathbf{cp}_j, \mathbf{n}_j, \theta_{p_j}\}_{j=1,\ldots,n}$, $\mathbf{cp}, \mathbf{n}, \theta_p$ define the pushing action, 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 effects, the push planning problem is how to iteratively select the push action based on the target direction and the push effects 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.

4.0.1 Pushing Primitive Representation

4.0.1.1 Action maps

The way of describing pushing primitives plays an essential role in modeling the push effect. Pushing primitives can be described by the pusher's starting and

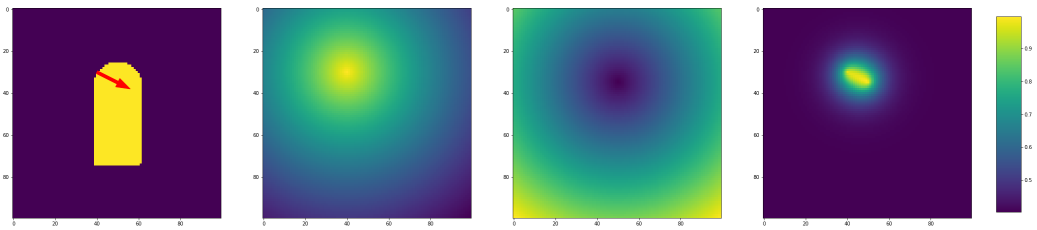


Figure 4.1: Action maps: The first one is the mask image and the applied pushing action, the middle two images are the action maps generated based on start pushing position and end pushing position respectively. The last image is to visualize the difference between action maps.

terminating positions [15, 18, 34, 67], 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 that provides better generalization capability to novel object shapes. Specifically, we use the distance transform formulation defined by Eqs. 4.1 through 4.3 to create two images with the same resolution as mask images called action maps. The term c in Eqs. 4.1 and 4.2 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. 4.1 shows an example of the generated action

maps and the visualization between those action maps.

$$s(x,y,x_s,y_s) = e^{-\frac{d(x,y,x_s,y_s)}{c}} \tag{4.1}$$

$$t(x,y,x_t,y_t) = \frac{d(x,y,x_t,y_t)}{c} \tag{4.2}$$

$$d(x,y,x_p,y_p) = \sqrt{(x-x_p)^2+(y-y_p)^2} \tag{4.3}$$

4.0.1.2 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. 4.2. 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 [68] 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.

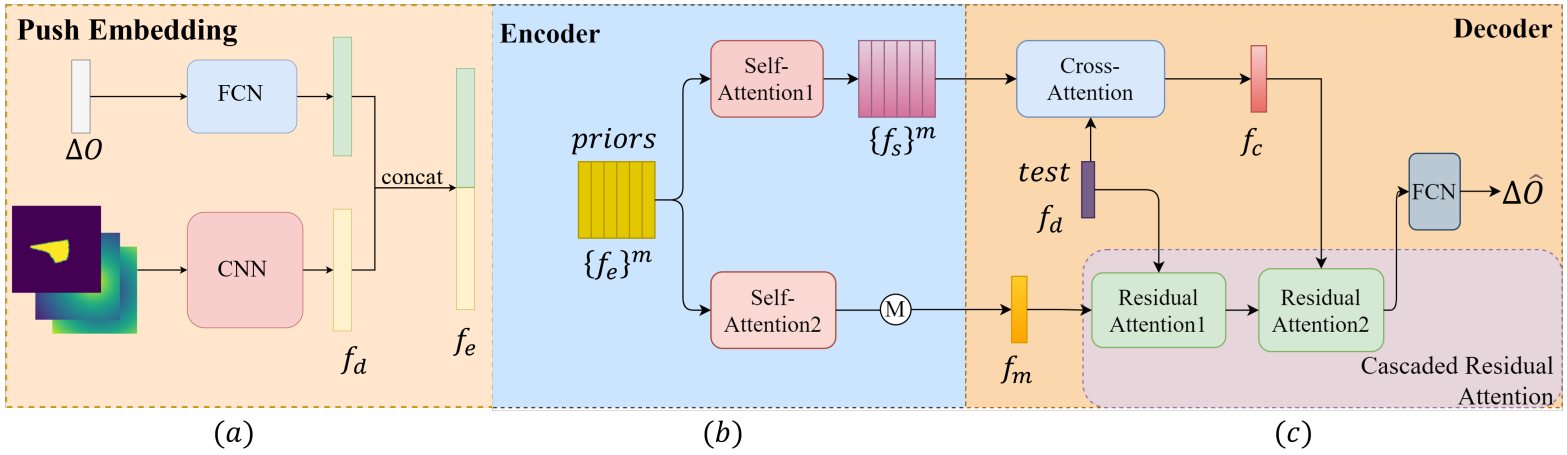


Figure 4.2: Proposed push embedding model (a) and encoder-decoder learning model (b and c). In push embedding, CNN takes the stacked object mask and action maps as input and encodes the spatial relations between the object state and action. FCN projects ΔO to the same dimensional space as f_d. Residual attention module is utilized to selectively combine them to output f_e.

4.0.2 Proposed Learning Model

We develop the push effect prediction module based on ANP [35] to model the causality between the applied pushing action and the resultant object motion. Fig. 4.2 shows the proposed push effect 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 residual 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

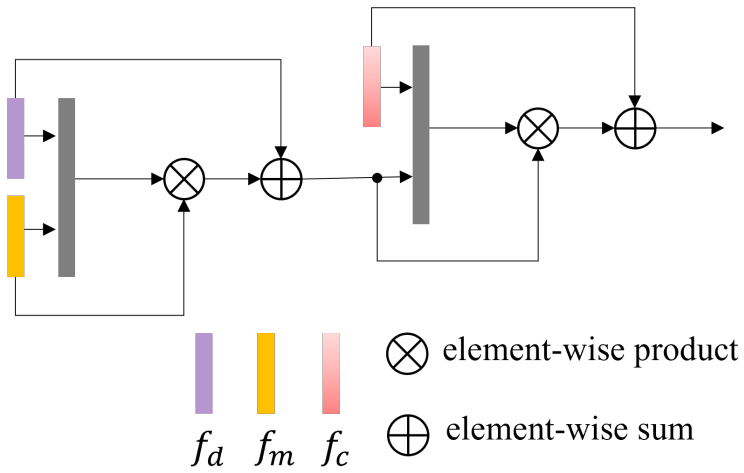


Figure 4.3: Cascaded residual attention module. the long gray blocks are the fully connected layers.

process can be represented by Eqs. 4.4 and 4.5.

$$F_{attn}(f_{in1}, f_{in2}) = tanh(concat(f_{in1}, f_{in2})W^T + b) \tag{4.4}$$

$$f_{out} = F_{attn}(f_{in1}, f_{in2}) \cdot f_{in2} + f_{in1}, \tag{4.5}$$

where *tanh* is the activation function. The cascaded residual attention model in decoder is shown in Fig. 4.3

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.

4.1 EXPERIMENTS

4.1.1 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.

4.1.2 Baseline and Ablation Models

4.1.2.1 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 [18].

4.1.2.2 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)** 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 selectively combines them and leads to more accurate pushing performance, instead of concatenating the input from a different source and feeding to MLP.
- **Model IV (single-pixel encoded action maps)** with the proposed model; the action maps are generated by creating an image with the same size as object mask, and it is initialized by assign all pixel intensities to zeros. One way to encode the pushing action is to discretize the pushing action into the pixel locations and assign the pixel intensities to one, however, discretization introduces extra error on prediction. Therefore, we apply linear interpolation to assign the pixel values to four neighbor pixel locations around the start and terminate pushing position.

4.1.3 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 [69] 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.

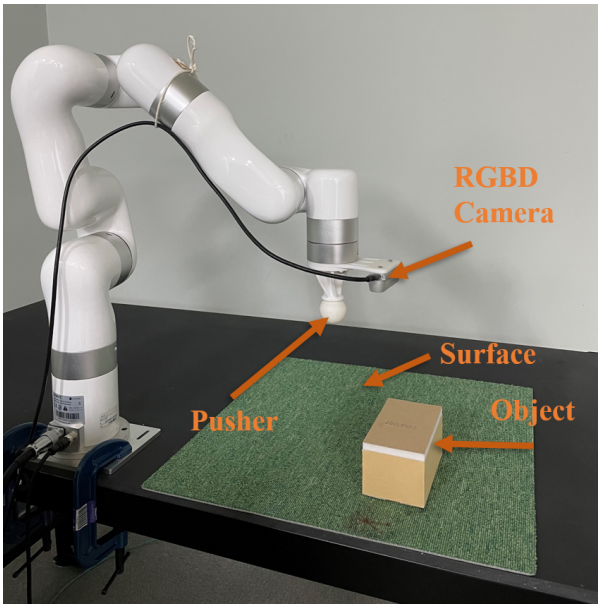


Figure 4.4: Real experimental setting.

4.1.4 Real Experiments

We evaluate the proposed object motion prediction model in an experimental setting using the XArm 5 Lite robot as shown in Fig. 4.4. 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 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 4.5. 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

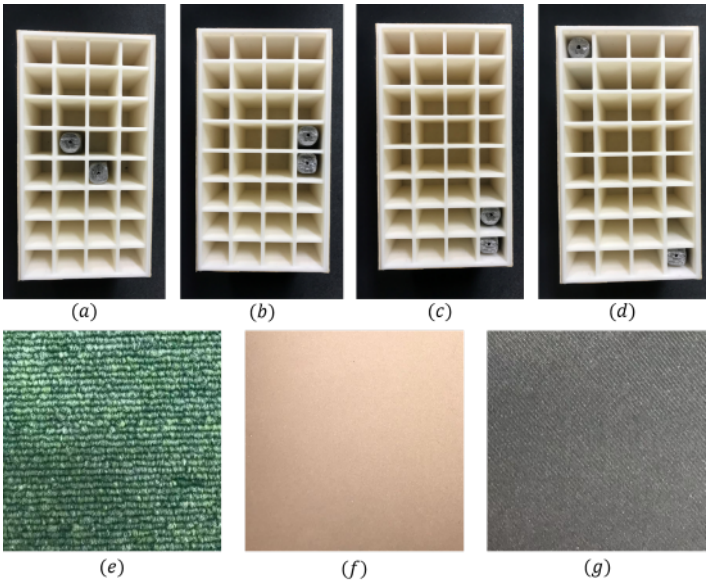


Figure 4.5: Different lead block positions ((a)-(d)) and surface frictions ((e) carpet, (f) foam, and (g) cloth)

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. 4.6. 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.

Figure 4.6: Real novel objects used in our experiment.

4.2 EXPERIMENTAL RESULTS AND DISCUSSION

4.2.1 Model Prediction Result

The performance of the models implemented on the test set are given in Table 4.1. 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 [18] are provided in Table 4.1. 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.

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. 4.1 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. 4.1.1 to prepare the test set and evaluate the proposed effect models to this test set. The result is shown in the last row of Table. 4.1. We observe that rotation prediction error becomes larger, however there is no difference in the translation prediction error.

To verify the influence of the number of available pushing priors to the prediction accuracy, we evaluate the trained model on test set with varied number of pushing priors. The range of the number of pushing priors available is from $(1, 12)$. The result is shown in Fig. 4.7. We observe that the mean and standard deviation of the prediction error get decreased with the increase of the number of pushing priors. On the other hand, the rate of improvement is decreasing with the increase of the number of pushing priors.

Table 4.2 and Table 4.3 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 4.1. However, the proposed model purely trained by the simulation dataset predicted pretty close. The average translation and rotation of

Table 4.1: 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
Model IV (single-pixel encoded action maps)	3.42	2.34	11.56	0.55	2.75	2.53	12.03	0.14
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 4.2: 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 4.3: 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

all the combinations are 20.8*mm* and 11.28°, and the mean prediction errors on translation and rotation are 4.17*mm* 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 4.4. The average translation and rotation of the objects are 23.13*mm*, 21.41*mm*, 19.72*mm*, 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.

4.3 Conclusion

Based on the collection simulation dataset and ANP, we proposed a novel method to encode pushes, which greatly improved the model performance. Based on the

Table 4.4: Prediction errors for objects motion in Fig. 4.6

	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

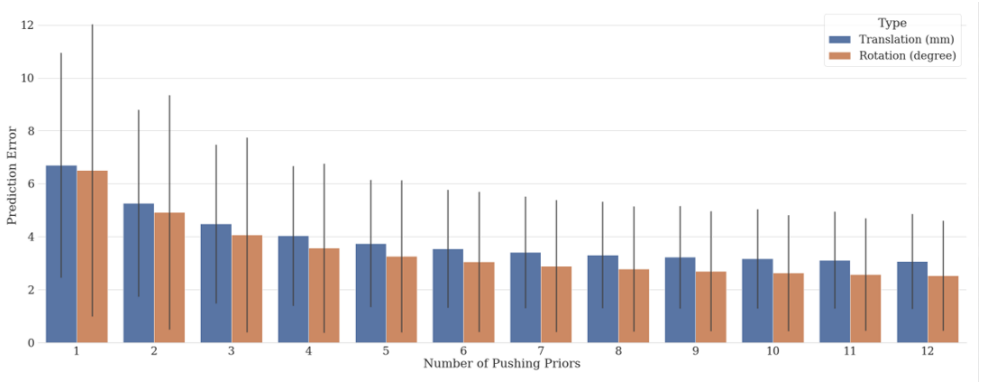


Figure 4.7: Prediction error with respect to different number of pushing priors.

encoder-decoder structure, we developed cascaded residual attention modules to combine features from different sources. Compared with ANP that concatenates all features together and feeds into MLP, our model efficiently combined the features and helped the representation learning.

Firstly, we evaluate the trained model on simulation test set, compared with the baseline models, our prediction model achieved the lowest mean and standard deviation of prediction error.

Then, we evaluate the trained model on larger magnitude pushing dataset. The result show that the trained model has similar performance on object translation but worse performance on object rotation. One reason is that the change in object orientation due to larger magnitude pushing is large and beyond the learned object motion distribution, which leads to a larger object rotation prediction error.

In addiction, we evaluate the trained model on test set with varied number of available pushing priors. Based on the result, we conclude that the more number of pushing priors available, the more accurate prediction. However, the rate of improvement is decreased with the increase of the number of pushing priors.

Finally, 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.

Chapter 5

Object Center of Mass Estimation

5.1 CoM Estimation by using the prediction model

In robotic grasping and manipulation, it is widely assumed that an object's inertial parameters are *a priori* known. The parameters can be estimated using a special hardware in a controlled environment, which may not be suitable for commonly used industrial robots.

Planar pushing can be used for object inertial parameter estimation by several pushing interactions with the object. Table. 5.1 represents several works on object inertial parameter estimation by planar pushing. In this work, instead of leveraging the force sensor or simulator, we propose a novel method to estimate the CoM of a novel object via planar pushing using only a position-controlled robot arm and a vision sensor available commercially off-the-shelf.

Table 5.1: Studies on object inertial parameter estimation

Method	**Force sensor**	**Simulator**	**Object**
Yu *et al* [55]	✔	✗	Box
Song *et al* [37]	✗	✔	Arbitray
Mavrakis *et al* [56]	✔	✗	Arbitray
Proposed	✗	✗	Arbitray

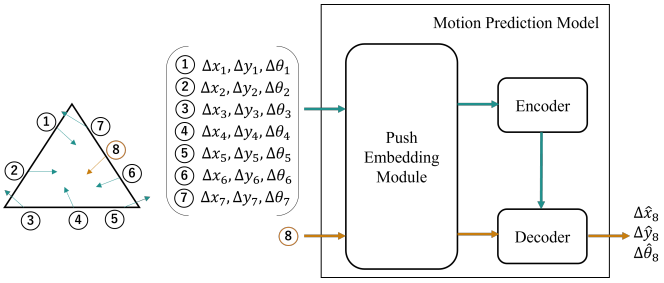


Figure 5.1: Schematic of object motion inference using motion prediction model.

In this chapter, we propose two kinds of method to CoM estimation by planar pushing. The first kind of methods combine our few-shot learning motion prediction model [70] and Mason's voting theorem (VT). Our learning model predicts the sense of rotation of a pushed object. The probable CoM region is selected through a decision process until it is narrowed down to the region whose centroid is considered the CoM of the object. The second kind of methods leverage dynamic pushing and voting theorem for estimating the object's CoM. Different from [56] which trains an end-to-end data-driven model for this task, our method explicitly reasons the effect of a pushing effect on narrowing the CoM region so that it can better generalize to novel object. In addition, our method can help to select pushing actions to efficiently interact the object so that to minimum the number of pushing needed.

In the following part, we review our few shot learning model and introduce the process of our CoM estimation method.

5.1.1 Prediction Model

Basically, the prediction model predicts object motion for a pushing primitive by integrating limited pushing priors based on the Residual Convolutional Network (ResNet) [68] and Attentive Neural Process (ANP) [35]. The specific inference procedure is as follows:

- Collect m pushing priors $\{\mathbf{cp}_i, \mathbf{n}_i, \theta_{p_i}, \Delta O_i\}_{i=1,\ldots,m}$, where $\mathbf{cp}_i$ is the contact location, $\mathbf{n}_i$ is the surface normal at $\mathbf{cp}_i$, θ_{p_i} is the pushing direction *w.r.t.* $\mathbf{n}_i$, and ΔO is the changes in object state represented by $\Delta x, \Delta y, \Delta\theta$.
- Feed the pushing priors to the pushing embedding model and encoder of few-shot learning model.
- Predict ΔO for test data.

An example is shown in Fig. 5.1, where a triangular object is subjected to pushing actions. There are seven pushes marked by the green arrows whose resultant object motions are known, and another push marked by the orange arrow whose resultant motion is unknown. Each push specifies a contact point **cp**, a normal direction **n**, and a push direction *w.r.t.* the normal direction θ_p. The pushes in green with its resultant motions form the pushing priors. The prediction model aims to predict the resultant motion for the push marked in the orange arrow by integrating the pushing priors. We denote the few-shot learning model by **F**

$$\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{\theta} = \mathbf{F}(\mathbf{cp}, \mathbf{n}, \theta_p, \mathbf{S}_{prior}) \tag{5.1}$$

where $\mathbf{S}_{prior}$ indicates pushing priors. As the few-shot learning model is used for the process of estimating the CoM, only $\Delta\hat{\theta}$ is used. Therefore, $\Delta\hat{x}$ and $\Delta\hat{y}$ are ignored in this work.

5.1.2 VT for CoM Estimation

VT relates the sense of rotation of a pushed object with the spatial relationship between several rays and the CoM of the object. VT states that three rays, R_L, R_R delimiting the left and right boundaries of the friction cone at the contact point and R_P indicating the line of pushing, vote for the sense of rotation. For instance, if any two of the three rays have a negative moment about the CoM, the object

would rotate clockwise regardless of the third ray generated moment.

The probable location of the CoM can be determined by analyzing the relationship between the rays and the resultant object rotation. A pushing example is illustrated in Fig. 5.2(a). The red arrow is the pushing direction specifying the ray R_P, when R_L and R_R are assumed to be known. If the object rotates clockwise, at least two rays have a negative moment about the CoM. Therefore, R_L and R_R must have a negative moment and R_P has an indeterminate moment about the CoM. Then, the probable CoM location can be determined by R_L and R_R. As R_L has negative moment to area two, three, and four, while R_R has negative moment to area three and four, area three and four are the areas to which both R_L and R_R have negative moment. Therefore, CoM must be located in area three and four. The VT cannot be used to locate the CoM of a novel object as R_L and R_R are unknown.

5.1.3 Region Selection Rules

We describe our method for narrowing the probable location of the CoM. Notably, a friction cone is not given at the contact point. The method determines the CoM location using the pushing direction and the sense of rotation of a pushed object. In our method, the sign of moment that R_P generates about the CoM is always determined. Therefore, the probable CoM location can be narrowed using R_P and the resultant rotation. An illustrative example is shown in Fig. 5.2.

In order to convey the idea clearly, we consider the following two cases. In the first case, the pushing direction is aligned with the contact normal as shown in Fig. 5.2(b), where R_P is inside the friction cone. In this case, if the sense of rotation is known, then the probable CoM region can be determined. Let us assume that R_P is aligned with the contact normal, and the object rotates clockwise. If the CoM of the object is outside the friction cone, both R_L and R_R should have a negative sign of moment about the CoM, and R_P must have the

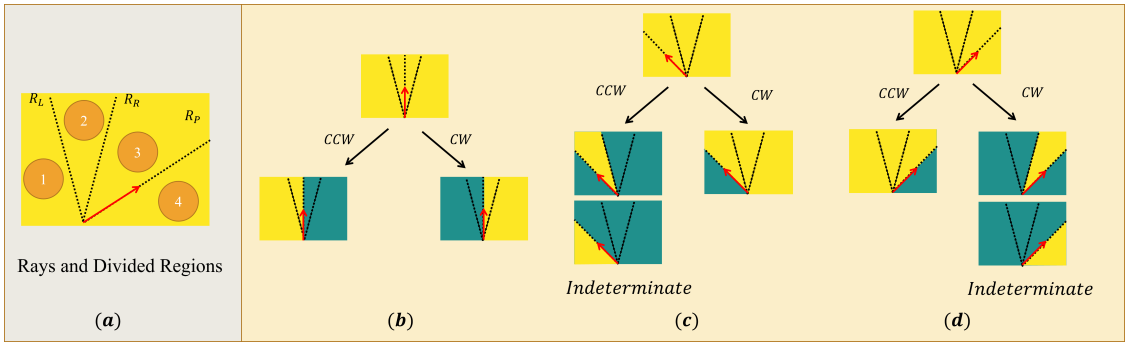


Figure 5.2: Probable CoM location narrowing: Yellow region is considered the probable CoM location, while green region is excluded. Red arrow is the pushing direction. Black dash lines are friction cone limits at the contact point. Probable location is initialized to the convex hull of the object mask. When pushing direction is aligned with the contact normal, the location can be narrowed if the sense of rotation is known. When pushing direction is not aligned with the contact normal, there are two cases that we cannot narrow the location. In such cases, we use another push action to continue to narrow the location.

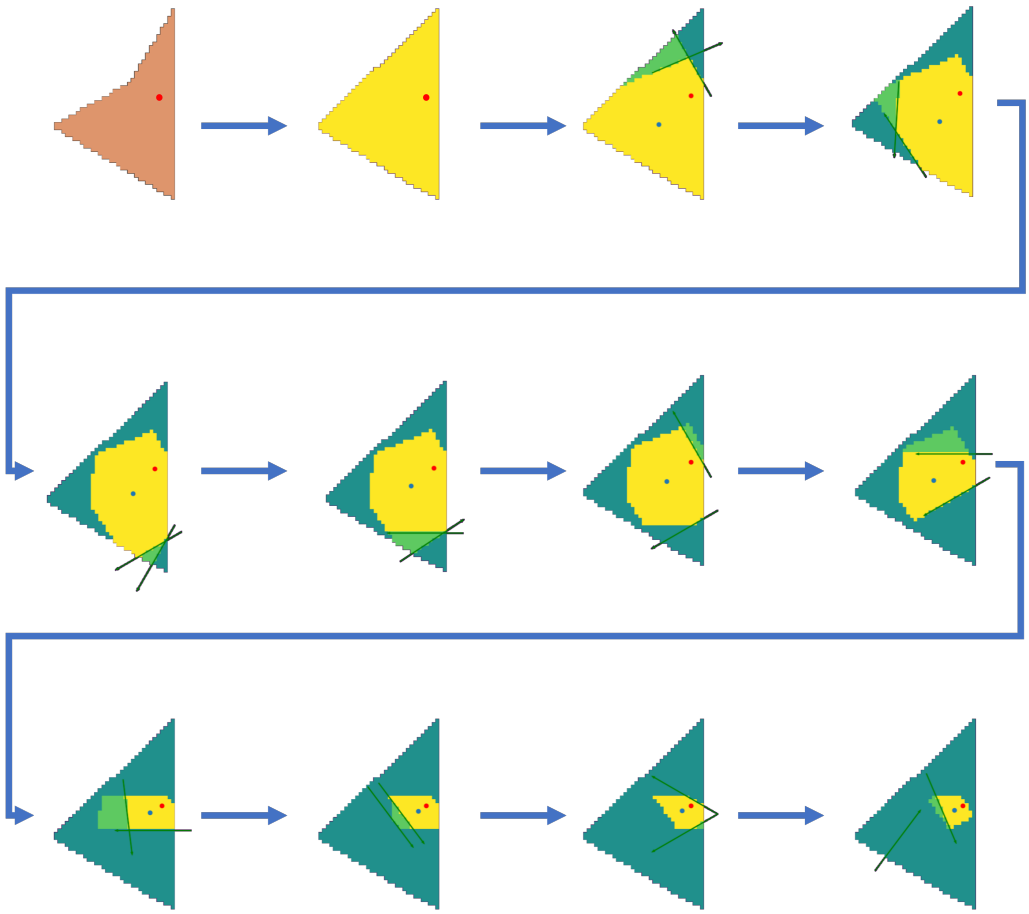


Figure 5.3: CoM estimation process. The first image illustrates the object with ground truth CoM in red dot. The second image shows the convex hull of the object as the probable CoM location in yellow color. Other images show a sequence of narrowing locations until the size reaches a threshold or no pushing action helps narrow the location.

Algorithm 2: CoM Region Decision Process

Input: $\mathbf{F}, \mathbf{S}_{prior}, \mathbf{A}_{CH}, \{\mathbf{cp}_i\}^m, \{\mathbf{n}_i\}^m, \{\theta_p\}^a, \theta_T, a_T$

/* $\mathbf{F}$ is the prediction model, $\mathbf{S}_{prior}$ is the pushing priors, $\mathbf{A}_{CH}$, which consists of a set of pixel locations, represents the region inside the convex hull of the object. $\{\theta_p\}^a$ is a set of angles w.r.t. normal direction, $\{\mathbf{cp}_i\}^m$, $\{\mathbf{n}_i\}^m$ are sampled contour point and normal direction associated. θ_T is a threshold for predicted rotation. a_T is a threshold of the area of probable CoM region. */

Output: $\mathbf{A}_{CoM}$

/* $\mathbf{A}_{CoM}$ is probable CoM Region which consists of a set of pixel coordinates $\{\mathbf{P}_{pix}\}$ */

$\mathbf{A}_{CoM} \leftarrow \mathbf{A}_{CH}$ // probable CoM region initialization

$\mathbf{S}_{push} \leftarrow \emptyset$ // initialize a cache

for θ_p *in* $\{\theta_p\}^a$ **do**

  for $\mathbf{cp}_i$ *in* $\{\mathbf{cp}_i\}_{i=1,...m}$ **do**

    $\hat{\theta}_i = \mathbf{F}(\mathbf{cp}_i, \mathbf{n}_i, \theta_p, \mathbf{S}_{prior})$ // rotation prediction

    if $\|\hat{\theta}_i\| > \theta_T$ **then**

      Append($\mathbf{S}_{push}$, $\{\mathbf{cp}_i, \mathbf{n}_i, \theta_p, \hat{\theta}_i\}$)

Sort $\mathbf{S}_{push}$ based on the amount of rotation in descending order

for $\{\mathbf{cp}_i, \mathbf{n}_i, \theta_p, \hat{\theta}_i\}$ *in* $\mathbf{S}_{push}$ **do**

  $\mathbf{A}_{CoM} \leftarrow$UPDATE($\mathbf{A}_{CoM}, \mathbf{cp}_i, \mathbf{n}_i, \theta_p, \hat{\theta}_i, a_T$)

same sign of moment with R_L and R_R as R_P is inside the friction cone. Then the regions where R_P has negative sign of moment can be regarded as the probable

Algorithm 3: Probable CoM Region Update

Input: $\mathbf{A}_{COM}, \mathbf{cp}_i, \mathbf{n}_i, \theta_p, \hat{\theta}_i, a_T$

Output: $\mathbf{A}_{CoM1}$

Function `Update` ($\mathbf{A}_{CoM}$, **cp**, **n**, θ_p, $\hat{\theta}$, a_T) **:**

- $\mathbf{A}_{CoM1} \leftarrow \mathbf{A}_{CoM}$ `// new probable CoM region initialization`
- Construct $\mathbf{R}_P$ based on **cp**, **n** and θ_p.
- **if** $\mathbf{A}_{CoM1}.size() >= a_T$ **then**
  - **return** $\mathbf{A}_{CoM1}$
- **else**
  - **if** $\mathbf{R}_P.intersect(\mathbf{A}_{CoM1})$ *is True* **then**
    - **if** $sign(\hat{\theta}) > 0$ **then**
      - **if** $\theta_p \geq 0$ **then**
        - **for** $\mathbf{P}_{pix}$ *in* $\mathbf{A}_{CoM1}$ **do**
          - `/* cross operation */`
          - **if** $\mathbf{R}_P \times \mathbf{P}_{pix} \leq 0$ **then**
            - Delete($\mathbf{A}_{CoM1}$, $\mathbf{P}_{pix}$)
        - **return** $\mathbf{A}_{CoM1}$
      - **else**
        - **return** $\mathbf{A}_{CoM1}$
    - **else**
      - **if** $\theta_p \leq 0$ **then**
        - **for** $\mathbf{P}_{Pix}$ *in* $\mathbf{A}_{CoM1}$ **do**
          - **if** $\mathbf{R}_P \times \mathbf{P}_{pix} \geq 0$ **then**
            - Delete($\mathbf{A}_{CoM1}$, $\mathbf{P}_{pix}$)
        - **return** $\mathbf{A}_{CoM1}$
      - **else**
        - **return** $\mathbf{A}_{CoM1}$
  - **else**
    - **return** $\mathbf{A}_{CoM1}$

CoM regions. If the CoM is between R_L and R_R, R_L and R_P must have a negative moment about the CoM, as there should be at least two rays that have a negative sign of moment about the CoM. In this case, the regions where R_P has a negative moment can be considered the probable CoM region.

The second case is that the pushing direction is different from the contact normal as shown in Fig. 5.2(c) and (d). This case introduces ambiguity as the sense of rotation might be dominated by R_L or R_R. Let us consider the left side of Fig. 5.2(c), where the object rotates counter-clockwise, and R_P can either have a positive or negative moment about the CoM. Therefore, the probable CoM region cannot be inferred by R_P. The same applies to the right side of (d) in Fig. 5.2. On the other hand, for the right side of Fig. 5.2(c), the object rotates clockwise, and R_P and R_L must have a negative sign of moment about the CoM. In this case, the CoM region can be narrowed by R_P and so is the left side of (d) in Fig. 5.2.

5.1.4 Combined CoM Estimation Method

We leverage the predicted sense of rotation of a pushed object to narrow the probable CoM regions in an iterative manner using Algorithm 2 and Algorithm 3. First, we collect a series of pushing examples as pushing priors. These pushing priors are used to help infer resultant object rotations for other pushes. Then, we sample a series of contact points uniformly around the object perimeter. Incorporated with the pushing priors, for each contact point, we query the prediction model to predict the rotation for different pushing directions at that contact point. Based on our experience, the deep learning model predicts accurate resultant object rotation directions if the pushing actions cause large amount of object rotations, but it is less accurate to predict for pushing actions which cause small amount of objects rotations. Therefore, we remove the pushing actions whose corresponding predicted amount of resultant object rotations are smaller than θ_T, and we sort

the pushing actions based on the corresponding predicted amount object rotations in descending order. After that, the sorted pushing actions with corresponding predicted object rotation directions are used to narrow down the probable CoM region until the ratio of the sizes between probable CoM region and the region covered the convex hull of the object is small than the area threshold a_T. Fig. 5.3 shows a sequence of snapshots of probable CoM regions being narrowed down.

5.1.5 CoM detection using predicted accumulated object rotation

Inspired by the fact that if the contact force passes through the CoM, then the object will be purely translated, otherwise, the object will rotate clock-wise or counter clock-wise based on VT, we propose an algorithm summarized in Alg. 4. It assumes given a set of actions defined by their start and end-points and their predicted motions on the object. Our idea is based on accumulating rotations for each point (pixel) in the object mask image weighted accordingly to their distance to lines defined by the start and end-point actions and identifying the region that has the lowest amount of accumulated rotation as a potential region where the CoM is possibly located. This can be also interpreted as the region where actions generating a small amount of rotations pass. For each action line (calculated via its start and end-points), we compute the distance from pixels to the line and use this distance as a weight for the predicted amount of rotation generated by this action. This way of the computed amount is accumulated for each pixel used for each action. We then apply a threshold to segment the potential region. Due to the selected threshold, there might be some isolated locations. In order to filter them out, we make use of the connected component method to find the largest connected region within the segmented pixels. This largest connected region is regarded as the potential CoM region and its center is computed as the estimated

Algorithm 4: CoM Estimation by Accumulated Motion

Input: Object Mask $\mathbf{M} = (x_i, y_i) \quad i = 1, 2, \cdots, m$,

List of actions **Action** and predicted rotation $\Delta\theta_i \quad i = 1, 2, \cdots, n$.

Area segmentation threshold $a_T \in [0, 1]$

Output: Predicted location of $\mathbf{P}_{CoM}$

/* Create an auxiliary matrix, $\mathbf{S}_\theta$, for accumulating distance weighted rotation amount */

$\mathbf{S}_\theta \leftarrow 0$

foreach *action in* **Action do**

  construct $\mathbf{R}_p$ based on action

  foreach $\mathbf{P}_{pix}$ *in* $\mathbf{M}$ **do**

    $d \leftarrow$ Compute distance between $\mathbf{P}_{pix}$ and $\mathbf{R}_p$

    $\mathbf{S}_\theta(\mathbf{P}_{pix}) \leftarrow \mathbf{S}_\theta(\mathbf{P}_{pix}) + e^{(-d)} \cdot |\Delta\theta_a|$

$\mathbf{S}_\theta \leftarrow$ normalize $\mathbf{S}_\theta$ to the interval $[0, 1]$

$\mathbf{A_c} \leftarrow S_\theta < a_T$ /* Create a segmented region of object mask as $\mathbf{A_c}$ */

$\mathbf{A_c} \leftarrow$ The largest connected region in $\mathbf{A_c}$

$\mathbf{P}_{CoM} \leftarrow$ Compute the centroid of $\mathbf{A_c}$

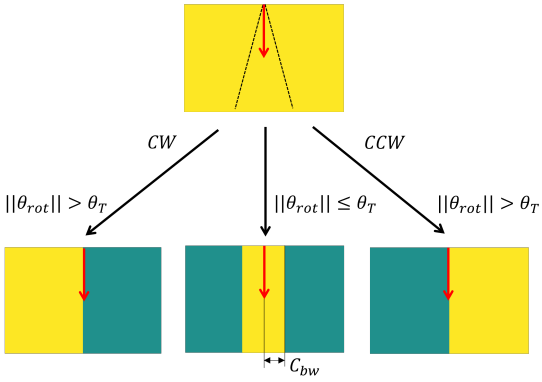


Figure 5.4: Estimating the CoM region: The red arrow in the direction of the contact normal represents the pushing action. The black dash lines delimit the friction cone. CW and CCW refer to clockwise and counter-clockwise rotations, respectively. The yellow regions represent the CoM region and the dark green regions do the non-CoM region.

CoM position.

5.1.6 CoM estimation by using Dynamic Pushing

In other methods proposed previously, we leveraged a prediction model to predict the resultant object rotations, then we use sampled pushing action and the associated predicted object rotation directions to narrow down the CoM region. However, there are two main shortcomings: the needs of pushing priors and prediction uncertainty. In this section, in stead of leveraging on prediction model, we estimate the object CoM online and minimize the number of pushes needed.

Our main idea is to constrain the pushing direction to the contact normal to ensure that R_P lies in the middle of the two boundaries of the friction cone. By doing so, the indeterminate cases shown in Fig. 5.2 (c,d) can be avoided. the CoM region can be uniquely determined by the spatial relationship of pushing direction as well as the CoM. In practice, there are some circumstances that the

object rotates so small that this amount cannot be easily detected by a vision sensor due to sensor limitations (*e.g.*, resolution or distance between the object and the sensor). Fortunately, in such cases, the distance between CoM and R_P is small. Therefore, we intuitively set a threshold θ_T for object rotation and a confidence bandwidth C_{bw} for possible CoM region selection. If the resultant object rotation is less than the threshold, the region whose inner pixel locations to R_P is less than the selected confidence bandwidth is regarded as the CoM region. Fig. 5.4 illustrates the selection rules for the proposed method.

The pseudocode of the proposed method is illustrated in **Algorithm 5** and **Algorithm 6**. The CoM region is initialized in **Algorithm 5** with the region inside the convex hull of the object mask. For each push, we use Principal Component Analysis of the current CoM region to find its centroid $\mathbf{c}$ and principal components $\mathbf{V}$. Then, we check if the corresponding line of each pushing action passes through the CoM region or not.

We compute the distance vector $\mathbf{d}$ between $\mathbf{c}$ and each line of pushing specified by the contact normal $\mathbf{N}_{ct}$. We then use a linear cost function to score each pushing action given by

$$s = \mathbf{w}^\top \begin{pmatrix} \mathbf{d}^\top \\ (\mathbf{N}_{ct}\mathbf{V}_1)^\top \end{pmatrix}, \tag{5.2}$$

where $\mathbf{w}$ is a two-dimensional weight vector and $\mathbf{V}_1$ is the main principal vector. As both $\mathbf{N}_{ct}$ and $\mathbf{V}_1$ are normalized, $(\mathbf{N}_{ct}\mathbf{V}_1)^\top$ represents the cosine similarity between $\mathbf{N}_{ct}$ and $\mathbf{V}_1$. We aim to find the pushing action that has a small cosine similarity to $\mathbf{V}_1$ to avoid a prolate-shaped CoM region. Using this cost function, the pushing action can be selected that has a close distance to $\mathbf{c}$ and small cosine similarity with $\mathbf{V}_1$. Following resultant object rotation, **Algorithm 6** updates the CoM region.

On the other hand, the assumption of Voting Theorem is quasi-static dynamics

Algorithm 5: CoM Region Decision Process by dynamic pushing

Input: $agent, \mathbf{P}_{ch}, \mathbf{P}_{ct}, \mathbf{N}_{ct}, \mathbf{w}, \theta_T, C_{bw}, n_{push}$

/* $\mathbf{P}_{ch}$, which consists of a set of pixel locations, represents the region inside the convex hull of the object. $\mathbf{P}_{ct}$, $\mathbf{N}_{ct}$ are the sampled contour points and normal directions associated. $\mathbf{P}_{ch}$, $\mathbf{P}_{ct}$, $\mathbf{N}_{ct}$ are all $(* \times 2)$ matrices, where $*$ is the number of sampled contour points or the size of region inside the convex hull of the object. n_{push} is the number of pushes that we set. */

Output: $\mathbf{P}_{CoM}$

/* $\mathbf{P}_{CoM}$ is probable CoM Region which consists of a set of pixel coordinates. */

$\mathbf{P}_{CoM} \leftarrow \mathbf{P}_{ch}$ // probable CoM region initialization

for $i = 1$ *to* n_{push} **do**

  /* opencv library */

  $\mathbf{c}, \mathbf{V}, \mathbf{x} \leftarrow PCA(\mathbf{P}_{CoM})$

  /* shapely library */

  $\mathbf{P}_{ct}, \mathbf{N}_{ct} \leftarrow valid(\mathbf{P}_{ct}, \mathbf{N}_{ct}, \mathbf{P}_{CoM})$

  /* calculate distances between $\mathbf{c}$ to the lines specified by $\mathbf{P}_{ct}$ and $\mathbf{N}_{ct}$ */

  $\mathbf{d} \leftarrow \frac{\mathbf{c}-\mathbf{P}_{ct}}{\|\mathbf{c}-\mathbf{P}_{ct}\|} \times \mathbf{N}_{ct}$

  /* Calculate scores for each contacts. */

  $$\mathbf{s} = \mathbf{w}^\top \begin{pmatrix} \mathbf{d}^\top \\ (\mathbf{N}_{ct}\mathbf{V}_1)^\top \end{pmatrix}$$

  j = argmin $\mathbf{s}$

  $\theta_{rot} \leftarrow agent.execute(\mathbf{P}_{ct_j}, \mathbf{N}_{ct_j})$

  $\mathbf{P}_{CoM} \leftarrow$UPDATE($\mathbf{P}_{CoM}$, $\mathbf{P}_{ct_j}$,$\mathbf{N}_{ct_j}$, θ_{rot}, c_{bw},θ_T)

Algorithm 6: Probable CoM Region Update of Alg. 5

Input: $\mathbf{P}_{CoM}$, $\mathbf{P}_{ct_j}$,$\mathbf{N}_{ct_j}$, θ_{rot}, c_{bw}, θ_T

Output: $\mathbf{P}_{CoM}$

Function `Update` ($\mathbf{P}_{CoM}$, $\mathbf{P}_{ct_j}$,$\mathbf{N}_{ct_j}$, θ_{rot}, c_{bw}, θ_T) **:**

- **if** $\|\theta_{rot}\| > \theta_T$ **then**
  - **if** $sign(\theta_{rot} > 0)$ **then**
    - **for** $\mathbf{P}_{CoM_i}$ *in* $\mathbf{P}_{CoM}$ **do**
      - **if** $\mathbf{N}_{ct_j} \times (\mathbf{P}_{CoM_i} - \mathbf{P}_{ct_j}) \leq 0$ **then**
        - Delete($\mathbf{P}_{CoM}$, $\mathbf{P}_{CoM_i}$)
    - **return** $\mathbf{P}_{CoM}$
  - **else**
    - **for** $\mathbf{P}_{CoM_i}$ *in* $\mathbf{P}_{CoM}$ **do**
      - **if** $\mathbf{N}_{ct_j} \times (\mathbf{P}_{CoM_i} - \mathbf{P}_{ct_j}) \geq 0$ **then**
        - Delete($\mathbf{P}_{CoM}$, $\mathbf{P}_{CoM_i}$)
    - **return** $\mathbf{P}_{CoM}$
- **else**
  - **for** $\mathbf{P}_{CoM_i}$ *in* $\mathbf{P}_{CoM}$ **do**
    - $d \leftarrow (\mathbf{P}_{CoM_i} - \mathbf{P}_{ct_j}) \times \mathbf{N}_{ct_j}$
    - **if** $d \geq C_{bw}$ **then**
      - Delete($\mathbf{P}_{CoM}$, $\mathbf{P}_{CoM_i}$)
  - **return** $\mathbf{P}_{CoM}$

and uniform ground where the slider slides. The CoM methods which leverage VT have several failure modes under quasi-static interactions.

- Sliding contact due to in-accurate estimation of contact normal.
- Pushing the object with decentralized density distribution.
- Non-uniform frictional ground.

In practice, the accurate contact normal may not be available when dealing with novel objects, in-accurate estimation on contact normal introduces estimation noise to the CoM region decision process. For instance, if the estimated contact normal is not bounded by the friction cone, when pushing the object along this estimated contact normal, the contact position will change due to the activation of sliding contact, which may results in reversed object rotation direction, then updating the CoM region by using this estimated contact normal and the resultant object rotation will lead to a wrong region which does not contain the CoM ground truth.

In quasi-static interaction, if the pusher pushes an object with decentralized density distribution, even the object rotation direction still obey VT, the amount of rotation is small so that it may not be observable to sensors.

The perfect-uniform ground may not be available in real setting, even it is that case, the physical properties of the ground will change dynamically because of the abrasion. non-uniform ground violate the assumption of Voting Theorem so that it results in in-accurate estimation on CoM.

In this section, we introduce the CoM estimation method by using dynamic pushing operation. Dynamic pushing operation means that the pusher pushes the object with high acceleration. One of the benefit of dynamic pushing is that it activates the effect of inertial force. The higher the acceleration of the pusher, the higher the inertial force is. With the help of inertial force, breaking contact occurs after terminating the movement of the pusher as the slider will keep moving due

to the inertial force. If the pusher pushes the slider with high acceleration and small motion magnitude, the period of the sliding contact will be small due to the breaking contact, which remove the noise introduced by the inaccurate contact normal estimation.

For pushing the object with decentralized density distribution. with the help of the inertial force, the resultant amount of rotation by dynamic pushing will be larger than the one of quasi-static pushing, which makes the rotation direction easier to be detectable.

By using dynamic pushing operation, the effect of frictional force generated by the ground to the object becomes smaller comparing with the contact force and inertial force. We argue that the CoM estimation by using dynamic pushing will be more robust than using quasi-static pushing.

5.2 EXPERIMENTAL SETTINGS

We carried out experiments both in simulation and real robot settings. We denote the surface friction coefficient as μ_s and the pusher friction coefficient as μ_p. We created a simulation environment composed of a plane and two spherical pushers. Coulomb's law of friction applies to the push-slider-supporting plane interaction and μ_s and μ_p can be adjusted. We used twenty different objects with convex and concave shapes in Fig. 7.5. The size of objects varies from $27cm^2$ to $150.5cm^2$.

A real experimental setup is shown in Fig. 5.5. A 5-DoF manipulator equipped with a variable opening stroke parallel-jaw gripper pushes a CoM changeable grid box on a flat surface. The stroke ranges from $0cm$ to $8.6cm$ and the size of the grid box is $14cm \times 8cm$. We wrap the gripper with two different materials: a low friction kraft paper and a high friction rubber. As for the plane surface, we test the table top surface itself and the foam surface. The grid box is wrapped in kraft

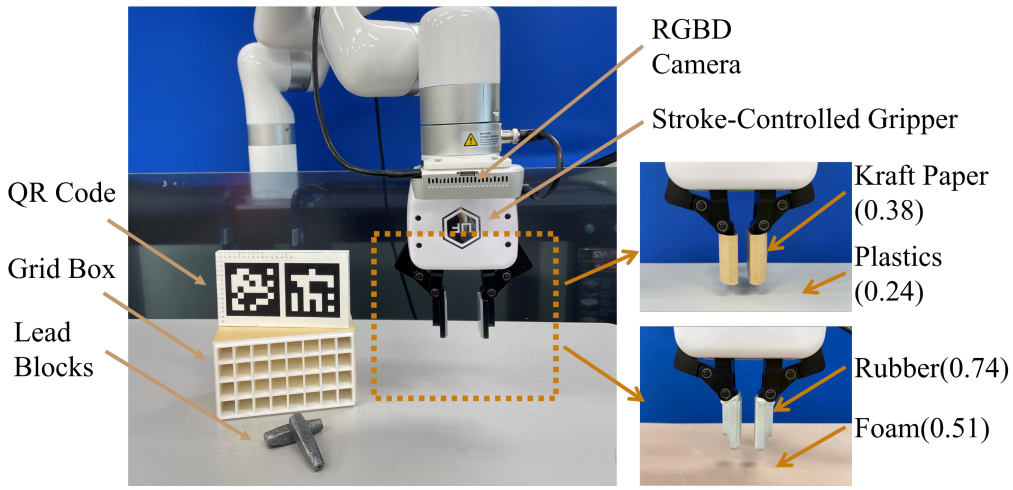


Figure 5.5: Real robotic pusher experimental platform. The left figure shows the CoM estimation and object translation experiment setting. The two right figures show the low and high frictional settings with the measured frictional coefficients in parentheses under the assumption of Coulomb's friction law.

paper. As the box has uniform density, the CoM is the same as its centroid. We put two lead blocks into different cells to change the CoM. The box weighs around 200 grams and each lead box has around 80 grams. The position and orientation of the box are monitored using ArUco Markers [71] attached to the top cover of the box.

5.3 CoM Estimation Experiment

5.3.1 Simulation Experiment

We evaluate the accuracy of our CoM estimation method in the following three aspects: (1) objects of different shapes and sizes, (2) different frictional settings, and (3) impact of the proposed deep learning model. For the first purpose, we use twenty different objects depicted in Fig. 7.5. For each object, we assign ten

different CoM locations. For the second purpose, we assign 0.1 and 1.0 for a low frictional setting and a high frictional setting, respectively, to both μ_s and μ_p. For the third purpose, as the prediction accuracy of the deep learning model highly depends on the pushing priors, we use different single contact pushing priors collected as follows. Given a specific object, we randomly sample several contact points. The robot pushes the object at each contact point $3cm$ forward along any of the $\{\theta_p\}^a$ directions. The contact points and pushing directions, as well as the resultant object motion are recorded as the pushing priors.

The pipeline is as follows: The pushing priors are collected for each object with a specified CoM location in each frictional setting. During pushing prior collection, We collect a total of twelve pushing priors for each estimation procedure. Then, the pushing priors are fed into the prediction model. As we set the total number of sampled contact points $\{\mathbf{cp}_i\}^m$ to fifty in Algorithm 2, and there is five direction pushing directions ($\{\theta_p\}^a$ is set to $\{-60°, -30°, 0°, 30°, 60°\}$.*w.r.t.* the normal direction at each contact point), the prediction model predicted resultant object motion for two hundred and fifty different pushing actions. All of the $\mathbf{R}_P$'s specified by the five pushing directions at each sampled contact point can divide the convex hull of the object into small enough regions to ensure that the estimation error will be bounded by the area threshold a_T to achieve the estimation accuracy requirement. The angle threshold θ_T is set to 2°, that is, if the predicted amount of rotation is less than 2°, we will not update the potential CoM region based on this pushing action. We set the area threshold a_T to 0.1, which means that the CoM estimation process will be terminated if the ratio between the sizes of the probable CoM region and convex hull of the object becomes smaller than 0.1. Finally, we run Algorithm 2 to find the potential CoM region and use its centroid as the estimated CoM. This CoM estimation pipeline is repeated three hundred times for each object with a specific CoM in each frictional setting. In

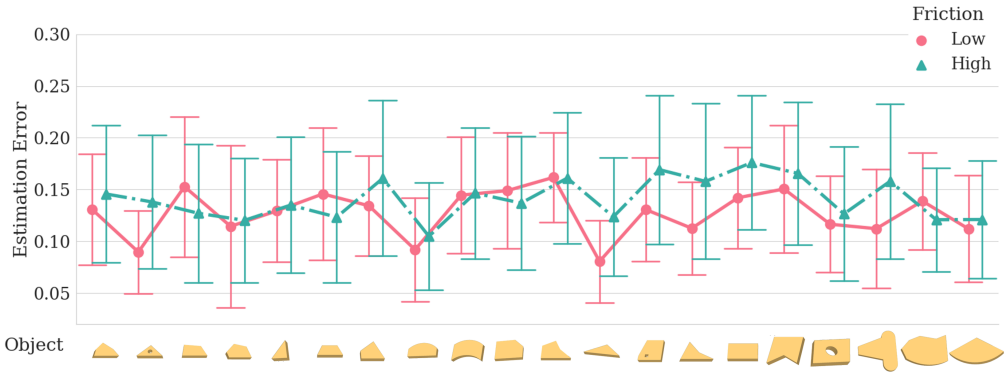


Figure 5.6: CoM Estimation Results in Simulation Environments. Objects are sorted by their representative size. Markers show the mean estimation errors and the intervals represent standard deviations. Red line shows the estimation error in low frictional setting and green dash-dotted line in high frictional setting.

each estimation procedure, different pushing priors are used. As we assign ten different CoMs and conduct CoM estimation in two friction settings, the CoM estimation was conducted six thousand times for each object.

The obtained results are presented in Fig. 5.6. The statistical results are computed over ten different CoM locations assigned to each object. The objects are sorted by the representative size (RS), defined by the distance between the object's centroid to the farthest point on its perimeter, and the estimation errors are multiplied by the reciprocal of RS. The main idea behind this is that if an object's physical properties are not observable, one can assume that the CoM is located at the centroid of that object. This would be same with the initial step in our algorithm before we apply any pushing action. The maximum error of the estimation occurs when the CoM ground truth is located on the object perimeter farthest from its centroid. Given that RS is the maximum estimation error, we provide the relative error computed separately for each object. As shown in Fig. 5.6, the mean estimation error is around 0.15. After investigating the effects

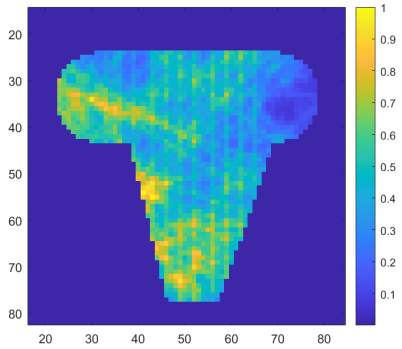


Figure 5.7: An example of normalized heat map representation for accumulated distance weighted amount of rotation using Object19.

of different frictional settings, we reached a conclusion that our CoM estimation method performs more accurately overall in a low frictional environment.

5.4 simulation experiment by predicted accumulated object movement

An example of a normalized heat map representation for accumulated distance weighted amount of rotation (represented by $\mathbf{S}_\theta$, which is computed in Alg. 4) is depicted in Fig. 5.7. Following this example, by applying a segmentation threshold and connected component algorithm, a potential CoM region can be determined. Fig. 5.8 denotes applying these steps to the example case.

Computational experiments were carried out on 300 different simulations for each object obtained using different parameters in object properties. Table 5.2 presents the statistical summaries (in the form of the mean and standard deviation) as well as the minimum and maximum distance between the real and the estimated CoM coordinates using Alg. 4. These distance measurements are given in *cm* and we also included the object sizes in cm^2.

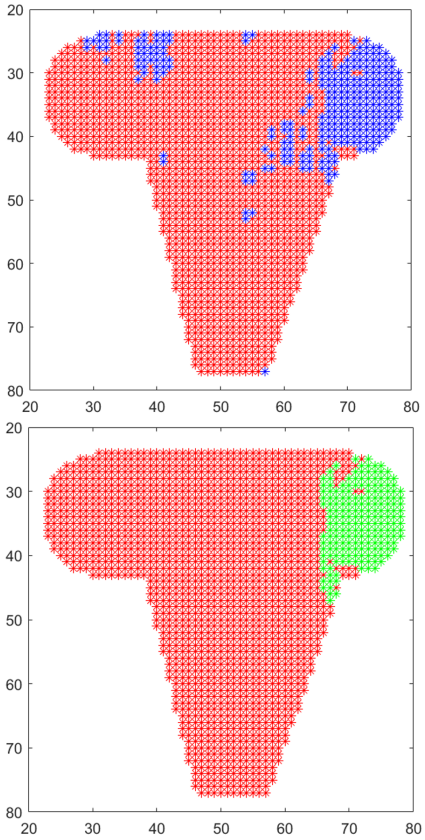


Figure 5.8: While the figure on top denotes segmented points, the figure bottom shows the largest connected component extracted from the segmented points. The center of this region is the estimated CoM and for this specific example, the distance between the estimated CoM and the real one is 2.02 pixels equivalent to 0.54cm.

Table 5.2: Computational results obtained using Alg. 4

Objects	Object Size	Low Friction				High Friction			
		mean	std	min	max	mean	std	min	max
Obj1	32.9	0.742	0.251	0.056	1.691	0.717	0.323	0.020	1.785
Obj2	35.6	0.696	0.317	0.056	1.693	0.697	0.327	0.020	1.950
Obj3	27.0	0.458	0.235	0.009	1.491	0.671	0.335	0.007	2.473
Obj4	103.3	1.004	0.381	0.019	2.076	1.311	0.515	0.005	3.213
Obj5	42.5	0.786	0.349	0.034	1.846	0.835	0.390	0.010	2.193
Obj6	58.0	0.856	0.335	0.008	2.293	1.034	0.397	0.021	2.412
Obj7	53.4	0.849	0.362	0.045	1.991	1.127	0.502	0.023	3.095
Obj8	110.4	1.096	0.407	0.047	2.471	1.223	0.510	0.065	3.084
Obj9	119.4	1.240	0.470	0.043	2.662	1.320	0.587	0.026	4.118
Obj10	91.0	0.906	0.427	0.013	2.030	1.462	0.536	0.034	2.786
Obj11	83.0	1.132	0.441	0.058	2.324	1.208	0.603	0.013	2.738
Obj12	58.3	1.100	0.361	0.050	2.463	1.075	0.461	0.022	3.643
Obj13	38.2	0.673	0.255	0.025	1.395	0.791	0.313	0.022	2.407
Obj14	43.0	0.859	0.440	0.066	2.185	0.749	0.344	0.015	2.153
Obj15	35.2	0.633	0.317	0.017	1.606	0.751	0.328	0.026	2.154
Obj16	67.8	0.939	0.323	0.079	3.673	0.884	0.378	0.016	2.352
Obj17	51.6	0.563	0.270	0.007	1.355	0.578	0.278	0.002	1.814
Obj18	120.4	1.029	0.445	0.043	2.801	1.170	0.591	0.012	4.210
Obj19	126.3	1.025	0.648	0.013	2.947	1.377	0.737	0.021	4.279
Obj20	150.5	1.233	0.485	0.173	3.329	1.317	0.561	0.101	4.133

5.5 Simulation experiment of CoM estimation by dynamic pushing

In this experiment, we compare the CoM estimation accuracy by using different pushing velocity. We used the same test objects used in the previous experiment. To examine the impact of the mass moment of inertia of an object about the z-axis, denoted by I_{zz}, we scale the default value of I_zz by a factor of 1 and 10. We use two different pushing speeds: one ($40cm/s$) is considered dynamic pushing, and the other ($4cm/s$) quasi-static pushing. In each run of CoM estimation, we execute 5 pushes, resulting in $6,400$ CoM estimations in total. Furthermore, to evaluate the performance on the anisotropic floor, we create a surface plane with two different frictional coefficients (0.1 and 1, respectively) along two principal axes, and set the pusher friction coefficient to 0.5. we follow **Algorithm 5** and set $\mathbf{w}$ to $[1, 0.5]^\top$. We set rotation threshold θ_T and confidence bandwidth C_{bw} to 1° and 10 pixels. The pushing length is set to $3cm$.

The simulation results of CoM estimation on isotropic frictional floor are detailed in Fig. 5.9 and Table 5.4. The result shows that the CoM estimation error with dynamic pushing is smaller than quasi-static pushing. We observed one notable case with quasi-static pushing that the estimation error tends to be large when an object has a large I_{zz}, or pushed across a high frictional floor, which results in a small amount of rotation misleading CoM region selection. On the other hand, in the case of dynamic pushing, as the wrench exerted by the floor on the object no longer dominates the object motion, the object motion follows the VT and the resultant rotation is larger than that of quasi-static pushing.

The simulation result of CoM estimation on anisotropic frictional floor is shown in Fig. 5.10 and Table 5.4. It can be observed that dynamic pushing still exhibits good performance even on the anistropic frictional floor. Compared with

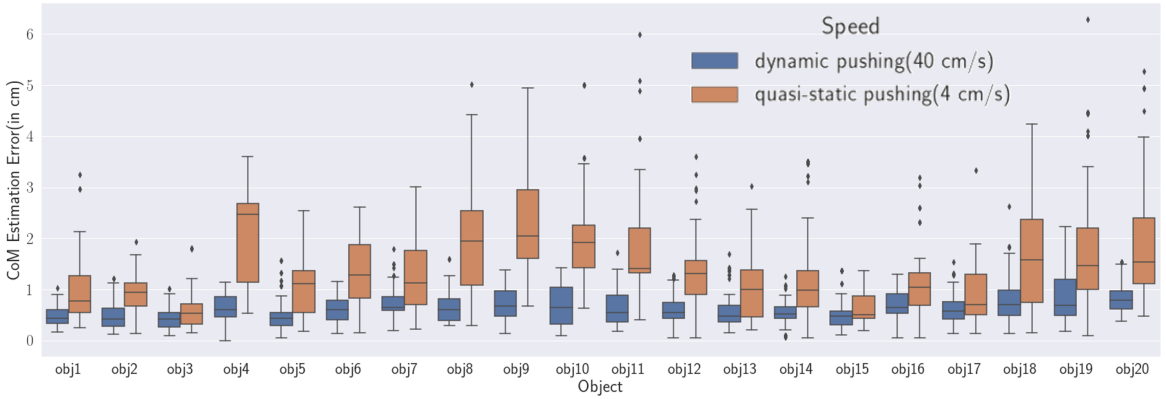


Figure 5.9: Experimental results on isotropic frictional floor.

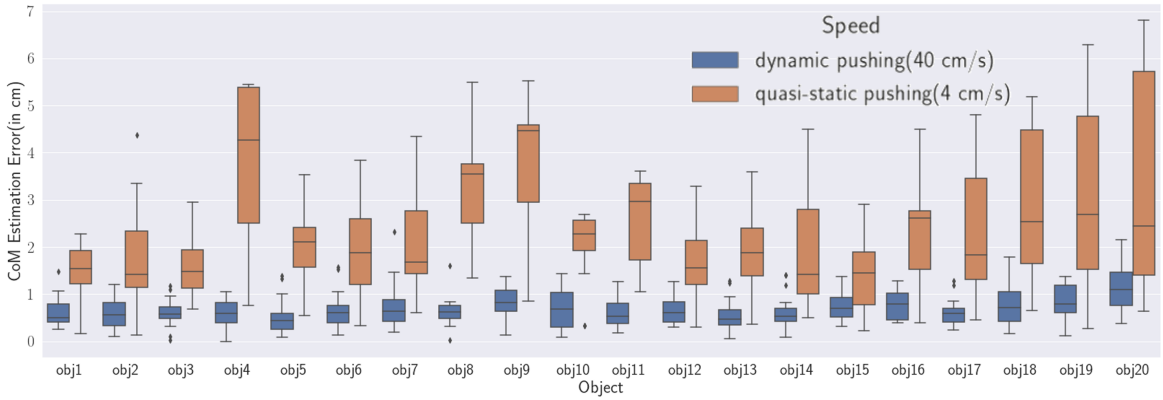


Figure 5.10: Experimental results on anisotropic frictional floor.

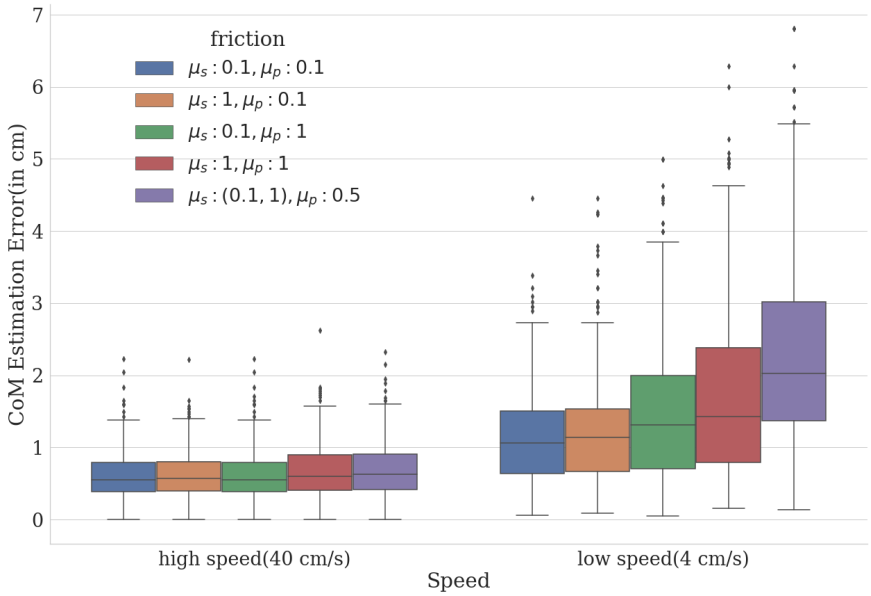


Figure 5.11: Robustness comparison between dynamic and quasi-static pushing.

the result on the isotropic frictional floor, the estimation error does not increase much. However, the estimation error with quasi-static pushing increases significantly. Fig. 5.11 shows the influence of friction on estimation error with dynamic and quasi-static pushing, respectively. Encouragingly, compared with the quasi-static pushing case, friction has a limited influence on estimation accuracy with dynamic pushing. The anisotropic frictional setting affects estimation accuracy small with dynamic pushing but deteriorates estimation accuracy significantly with quasi-static pushing.

The results showed that dynamic pushing can improve the accuracy and robustness of estimation, supported by a series of simulation experiments. As a result, the proposed CoM estimation method with dynamic pushing achieved impressive performance on CoM estimation for novel objects both in isotropic and anisopropic frictional floors.

Table 5.3: Results of CoM Estimation on Isotropic Floor

Object	**High Speed Pushing**				**Low Speed Pushing**			
	mean	**std**	**max**	**min**	**mean**	**std**	**max**	**min**
obj1	0.489	0.206	1.034	0.166	0.949	0.657	3.251	0.255
obj2	0.476	0.278	1.208	0.123	0.915	0.364	1.932	0.137
obj3	0.423	0.205	1.008	0.091	0.591	0.362	1.804	0.15
obj4	0.631	0.29	1.136	0.0	2.06	0.844	3.604	0.536
obj5	0.481	0.351	1.568	0.05	1.071	0.553	2.537	0.176
obj6	0.605	0.274	1.154	0.134	1.297	0.599	2.611	0.149
obj7	0.713	0.315	1.789	0.196	1.3	0.673	3.015	0.217
obj8	0.651	0.303	1.597	0.299	1.936	1.174	5.018	0.299
obj9	0.742	0.32	1.379	0.134	2.2	1.177	4.939	0.67
obj10	0.66	0.389	1.43	0.092	2.067	0.963	4.999	0.625
obj11	0.619	0.342	1.723	0.174	1.805	1.031	5.999	0.402
obj12	0.619	0.291	1.285	0.054	1.423	0.7	3.599	0.054
obj13	0.598	0.365	1.695	0.153	1.004	0.602	3.027	0.203
obj14	0.53	0.258	1.252	0.064	1.253	0.901	3.502	0.052
obj15	0.521	0.28	1.367	0.11	0.678	0.356	1.368	0.187
obj16	0.689	0.274	1.292	0.054	1.103	0.603	3.187	0.054
obj17	0.645	0.309	1.533	0.136	0.919	0.567	3.332	0.136
obj18	0.8	0.472	2.624	0.143	1.735	1.132	4.238	0.155
obj19	0.87	0.532	2.227	0.176	1.811	1.17	6.288	0.088
obj20	0.812	0.282	1.542	0.377	1.862	1.079	5.276	0.476

Table 5.4: Results of CoM Estimation on Anisotropic Floor

	High Speed Pushing				**Low Speed Pushing**			
	mean	**std**	**max**	**min**	**mean**	**std**	**max**	**min**
obj1	0.597	0.296	1.478	0.255	1.408	0.667	2.283	0.166
obj2	0.595	0.351	1.208	0.097	1.744	1.014	4.382	0.137
obj3	0.607	0.284	1.173	0.019	1.602	0.644	2.953	0.688
obj4	0.598	0.295	1.046	0.0	3.651	1.836	5.439	0.758
obj5	0.51	0.365	1.384	0.086	2.039	0.818	3.536	0.552
obj6	0.649	0.38	1.576	0.134	2.006	0.93	3.835	0.34
obj7	0.742	0.483	2.323	0.196	2.098	1.115	4.347	0.615
obj8	0.631	0.303	1.597	0.027	3.22	1.184	5.488	1.349
obj9	0.832	0.324	1.379	0.134	3.789	1.413	5.517	0.858
obj10	0.667	0.407	1.43	0.092	2.073	0.676	2.697	0.334
obj11	0.606	0.32	1.266	0.174	2.513	0.937	3.608	1.058
obj12	0.655	0.288	1.26	0.299	1.76	0.929	3.294	0.308
obj13	0.556	0.328	1.28	0.05	1.938	0.881	3.601	0.364
obj14	0.596	0.363	1.399	0.089	1.83	1.174	4.493	0.502
obj15	0.771	0.321	1.368	0.31	1.347	0.68	2.898	0.232
obj16	0.764	0.296	1.286	0.388	2.525	1.126	4.502	0.388
obj17	0.599	0.272	1.288	0.233	2.428	1.479	4.809	0.45
obj18	0.832	0.456	1.787	0.164	2.863	1.532	5.179	0.649
obj19	0.822	0.381	1.376	0.12	3.026	1.756	6.287	0.267
obj20	1.15	0.491	2.148	0.383	3.319	2.189	6.807	0.641

5.5.1 Experiments in Real Platform

In real experiments, we insert two lead blocks into the box to change the CoM, $\mathbf{P}_{CoM}$, that can be computed by Eq. 5.3.

$$\mathbf{P}_{CoM} = \frac{1}{M}\Sigma_i^n m_i \mathbf{r}_i, \tag{5.3}$$

where M is the total mass equal to the sum of masses of the box and two lead blocks, denoted by m_i, respectively, and $\mathbf{r}_i$ represents the position of m_i in a reference coordinate frame in the plane. We measured the masses of the empty box and each lead block using an electric scale with precision 0.1 gram. The $\mathbf{r}_i$ coordinate of the empty box was regarded as the centroid of the box as it has uniform density, and the $\mathbf{r}_i$ value of each lead block approximated to the center of the compartment in which each block was inserted.

We follow the previous pushing prior collection procedure in two different settings: a foam surface and the rubber wrapped gripper referred to as high frictional setting and a plastic surface and the kraft paper wrapped gripper referred to as low frictional setting. For each frictional setting, we select five different lead block cells as shown in Fig. 5.12, resulting in five different CoMs. We sample twelve contact points uniformly across the box perimeter, and the robot pushes each contact point along five directions. The grid box was pushed sixty times for each CoM configuration and frictional setting. The CoM estimation pipeline is the same as the simulation. The estimation procedure was repeated six hundred times, for each of which, twelve pushing priors are sampled from sixty pushing examples.

We use the same method to represent the estimation error in the real experiments of CoM estimation, which is shown in Fig. 5.13. It can be observed from the figure that our CoM estimation method can be generalized to real settings, comparable to the results of simulation experiments. Overall, the estimation

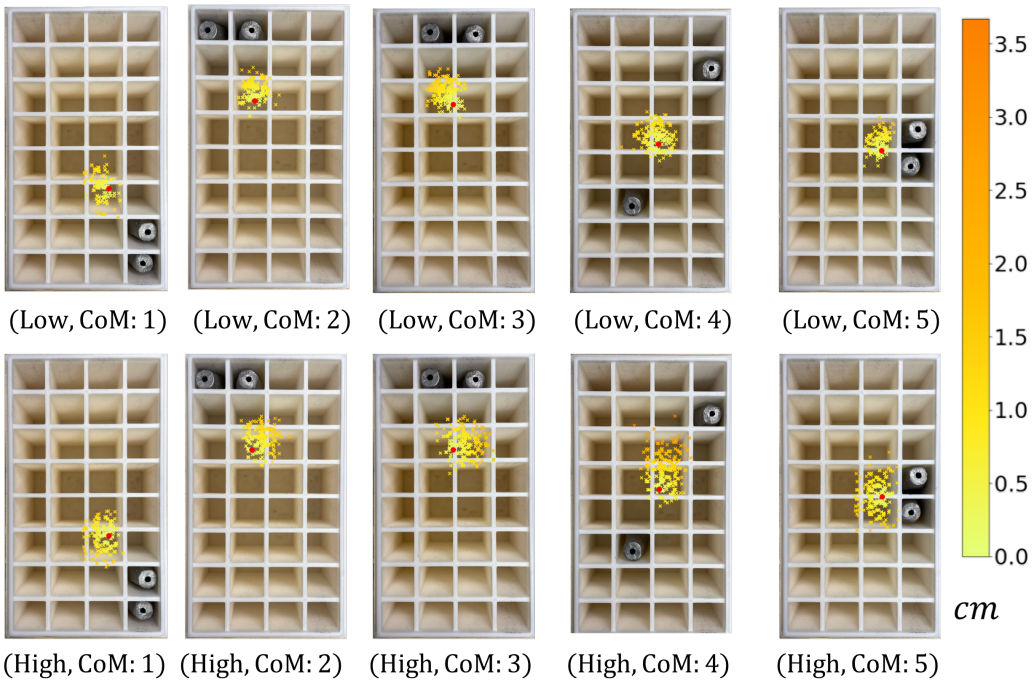


Figure 5.12: CoM estimation results in low and high friction settings. Red dot represents the ground truth CoM. Other dots are estimated CoMs. The brighter the color, the closer to the ground truth.

uncertainty is smaller in low frictional setting. As the result for the fourth CoM configuration, the estimation error became large. One reason is that, compared with other CoM configurations, the pressure distribution is more decentralized. Such a configuration causes an object to rotate less, leading to inaccurate CoM estimates.

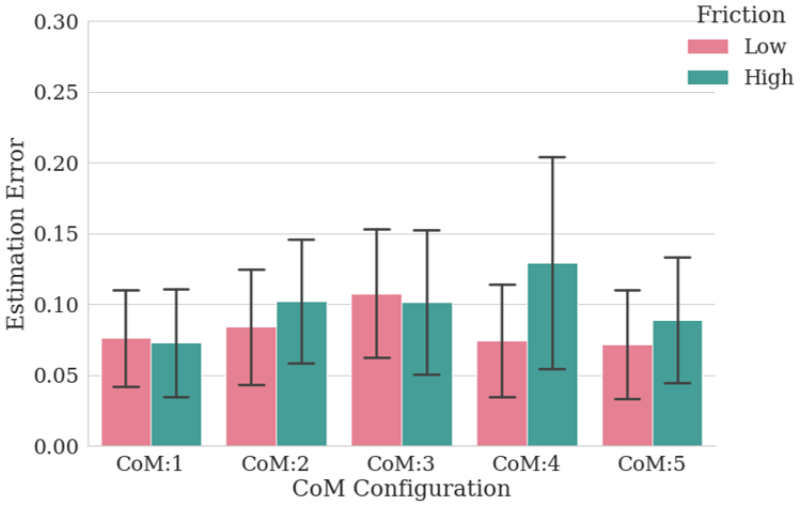


Figure 5.13: CoM Estimation Result in Real Experiment. Red bars show the estimation errors in percentage in low frictional setting, and green bars show the estimation errors in percentage in high frictional setting.

Chapter 6

Single-Contact Push Planning Using Data-Driven Model

6.0.1 Planning

Planning is to push the object from its initial pose to the target within pre-specified steps as shown in Fig. 6.1. 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 effect 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 effect 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 [15]. Then the push effect map is generated using the prediction model to predict the push effect for all representative actions.

In the phase of Efficient Action Selection in Fig. 6.1, 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

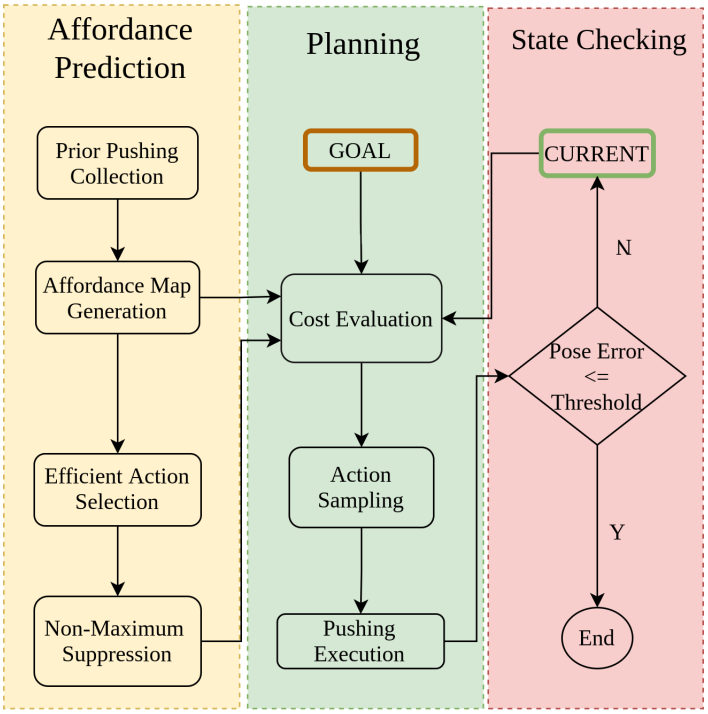


Figure 6.1: Pushing planning flowchart given the object's initial and target poses. A few prior pushes are collected for encoding the object dynamics. Then, the push effect map is generated by the trained model to predict the effect of the representative actions. After that, the representative actions 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.

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. 6.1. 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. 6.2 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. 6.2. $\mathbf{v}_p(a_i)$ 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. ${}^t a^*$ 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. 6.3. 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 ${}^t a^*$ after executing a_i using Eq. 6.4. $\theta_p(a_i)$ in Eq. 6.4 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. 6.1. The first term is the same as Eq. 6.3. The second term is the 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 ${}^t a^*$. 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

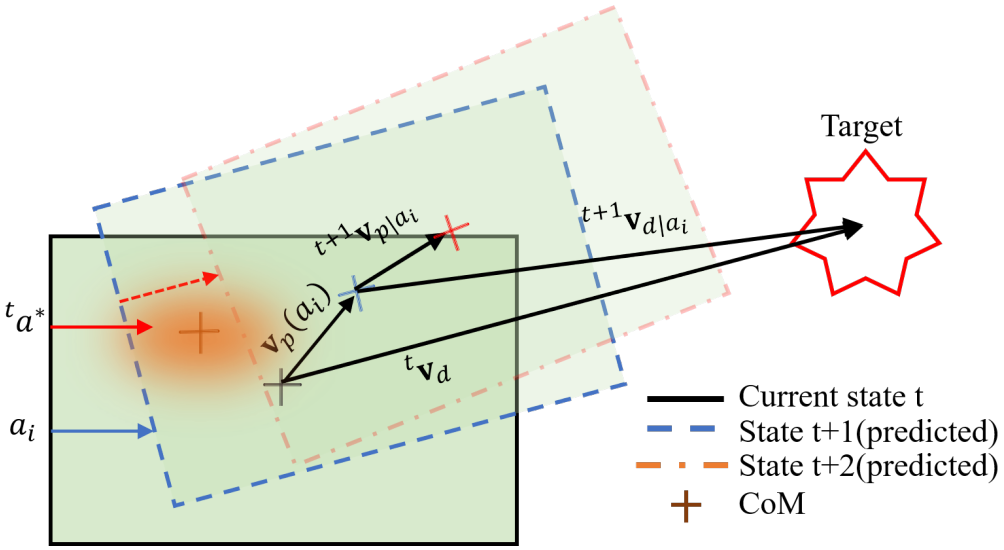


Figure 6.2: Illustration of the proposed planning method. a_i is the pushing action applied to the object at the current time step t. $\mathbf{v}_p(a_i)$ and $^t\mathbf{v}_d$ are the predicted and expected object displacement, respectively, at t. $^ta^*$ is the action selected based on Eq. 6.2. $^{t+1}\mathbf{v}_{d|a_i}$ is the expected object displacement at $t+1$ time step, when applying a_i calculated by Eq. 6.3. $^{t+1}\mathbf{v}_{p|a_i}$ is the predicted object displacement, when applying $^ta^*$ at $t+1$ time step by Eq. 6.4.

.

number of pushing steps exceed the requirement.

$$\phi(a_i) = \|^{t+1}\mathbf{v}_{d|a_i}\| - \lambda \frac{^{t+1}\mathbf{v}_{p|a_i} \cdot ^{t+1}\mathbf{v}_{d|a_i}}{\|^{t+1}\mathbf{v}_{d|a_i}\| \|^{t+1}\mathbf{v}_{p|a_i}\|} \tag{6.1}$$

$$^ta^* = \arg \max_{a_i \in \{a^*\}^m} \mathbf{v}_p(a_i) \cdot ^t\mathbf{v}_d \tag{6.2}$$

$$^{t+1}\mathbf{v}_{d|a_i} = ^t\mathbf{v}_d - \mathbf{v}_p(a_i) \tag{6.3}$$

$$^{t+1}\mathbf{v}_{p|a_i} = \mathbf{R}(\theta_p(a_i)) \cdot \mathbf{v}_p(^ta^*) \tag{6.4}$$

6.1 Experiment

6.1.1 planning in simulation

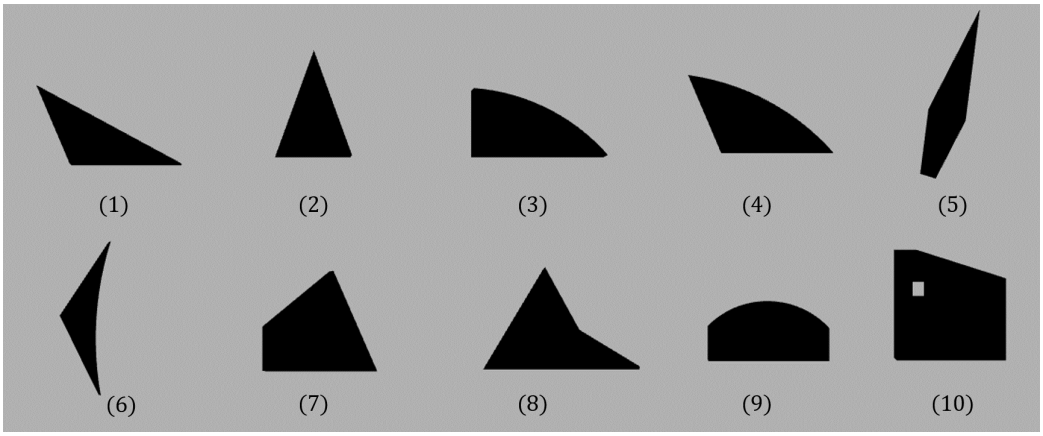


Figure 6.3: Test objects used for planning simulation.

Firstly, we evaluate the proposed planning method in the simulation environment using 10 different objects shown in Fig. 6.3, 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

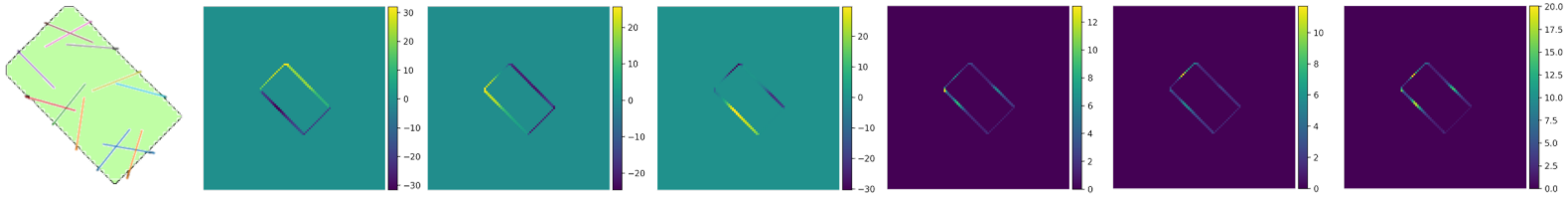


Figure 6.4: Illustration of pushes: (a) encoding object dynamics, (b)-(g) generated push effect map. Twelve pushes are conducted around the contour of the object along with a random direction within (-60, 60) *w.r.t.* the surface normal. (b) to (d) are the prediction object motion (translation and rotation) 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.

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. 6.4 illustrates an example of the push effect map generation procedure mentioned in Fig. 6.1. 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 6.0.1, we used 5 different pushing directions *w.r.t.* the surface normal vector. We used 500 representative pushing actions in total, and the push effects can be efficiently calculated on GPU in less than 1 second. Based on the push effects and uncertainty predicted, and the top 100 representative action candidates, we 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. 6.1 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 effect 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. 6.1. In other words, the baseline method only select actions that can minimize the translation error from the current to target positions.

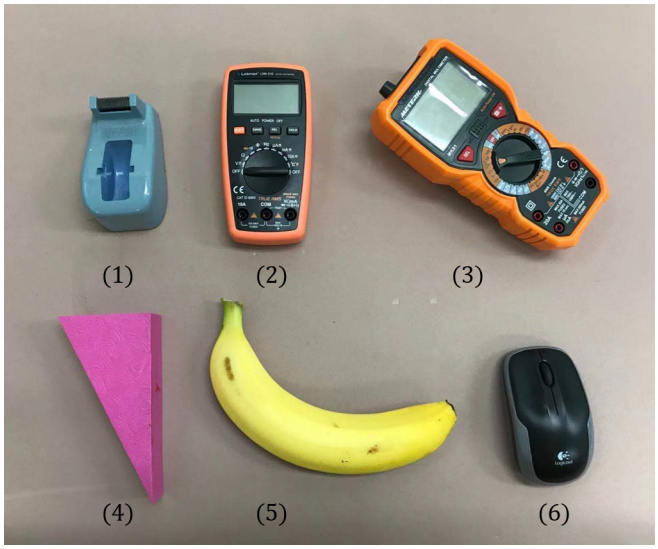


Figure 6.5: Test objects used for planning on real experiment setting.

6.1.2 planning on real platform

We evaluated the proposed method in real experimental settings on the foam surface with isotropic friction. We consider 6 different objects given in Fig. 6.5. 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.

6.1.3 Planning Evaluation

[ht]

The comparative performance of the proposed planning and the baseline methods is presented in Table 6.1 with the mean, standard deviation and accuracy of

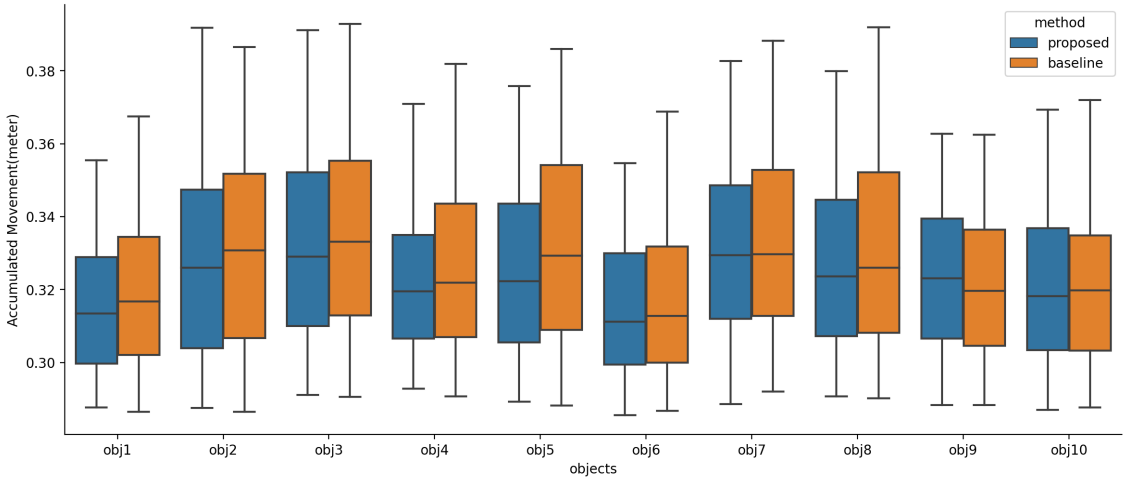
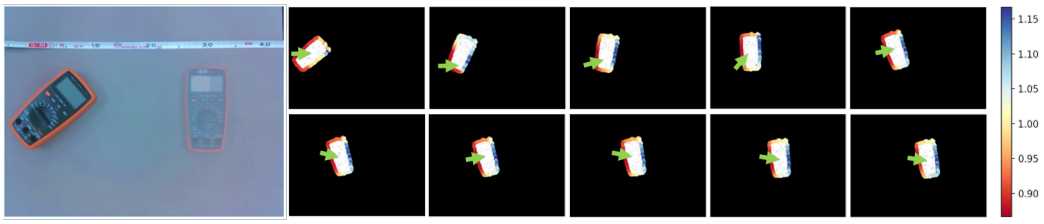


Figure 6.6: Accumulated movement of objects shown in Fig. 6.3 for the planning task. The line inside each color box is the mean of the accumulated movement.



(a) (b)

Figure 6.7: An example of pushing *test-object-exp* 3 in Fig. 6.5 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.

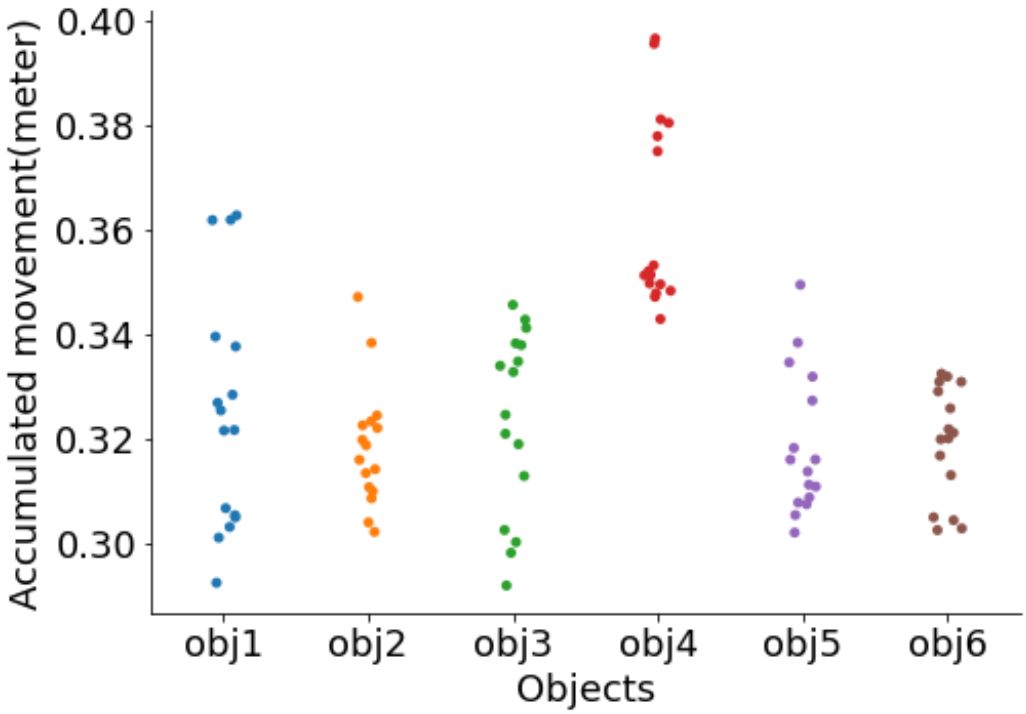


Figure 6.8: Accumulated movement of objects shown in Fig. 6.5.

pushing steps. Fig. 6.6 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. 6.3. From the results shown in Table 6.1, 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 relatively 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. 6.8 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 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. 6.7 shows an example of pushing *test-object-exp* 3 in Fig 6.5. 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.

Table 6.1: The mean, standard deviation of the pushing steps as well as the accuracy for objects shown in Fig. 6.3

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

Chapter 7

Multi-Contact Pushing For Novel Object

7.1 Two-Edge-Contact Pushing

We introduce our ZMTEP method for translating an object to a goal pose under the quasi-static assumption. First, the forces exerting on the object are analyzed when the object undergoes pure translation. We then revisit moment labeling representation for contact forces. We compare different contact configurations and analyze tolerance to noise on contact and estimated CoM positions. Finally, we present the ZMTEP method for two-edge contact configuration selection.

7.1.1 Quasi-Static Analysis of Pure Translation

Let us consider a pusher and an object lying on a flat surface with isotropic friction, where the CoM, the centroid of pressure distribution, and the centroid of friction are projected to the same point on the plane [72]. The magnitude of the contact force between the pusher and the object can be arbitrarily large due to the non-penetration constraint of a rigid body. In case the object moves in low speed, the inertia force would be in the order of the friction force. In translational motion, sliding friction is replaced by an equivalent resultant force passing through the

CoM that opposes the motion of the object.

As the two-edge-contact pushing exploits the pusher's collaborative contact interactions with the object, the direction of the net force must be the same as the pushing direction to achieve translational motion. The pusher's contact force acts inward on the object, and two contact forces must be able to set the object in motion toward the direction of the net force additively. Also, the line of the net force must pass through the CoM of the object to generate zero moment about the CoM.

7.1.2 Zero Moment Two Edge Contact Pushing (ZMTEP)

We first define several explanatory lines of direction to convey the proposed idea of two-edge-contact configuration. We then analyze the tolerance range of the contact configuration, leading to the ZMTEP for pure translation.

7.1.2.1 Defined Lines

We define five lines: $l_d, l_n, l_{fl}, l_{fr}, l_c$. l_d is the line passing through the CoM of an object and is aligned with the pushing direction. l_n is perpendicular to the surface at the contact point between the pusher and the object. l_{fl} and l_{fr} are the left and right boundary of the friction cone at the contact point, respectively. l_c is the direction of the contact force bounded by l_{fr} and l_{fl}. When there is no friction between the pusher and the object, l_c coincides with l_n. Fig. 7.1 illustrates an example for pushing an object along l_d, where two contact points between the object and the pusher are denoted by the subscript 1 and 2, respectively. The shaded region is the moment representation for this contact configuration, which will be introduced latter.

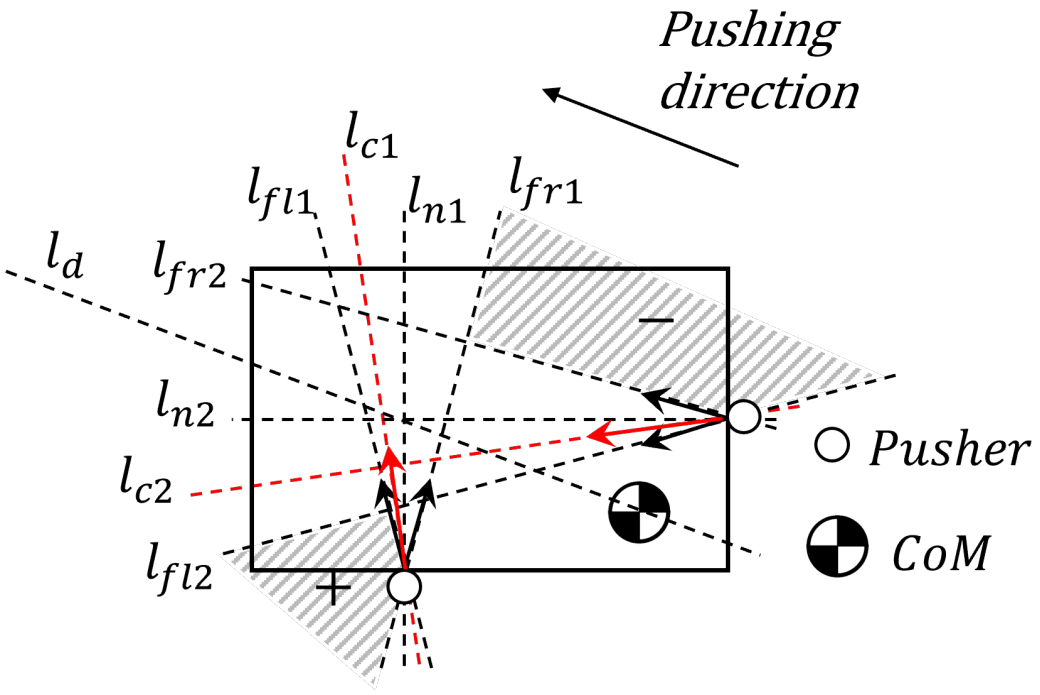


Figure 7.1: Illustration of the lines defined in 7.1.2.1.

7.1.2.2 Two-Edge Contact Configuration

Within the scope of two-edge-contact pushing between the pusher and the object, the contact configuration can be described by $\mathbf{C}_{tec}$

$$\mathbf{C}_{tec} = (\mathbf{cp}_1, \mathbf{n}_1, \mathbf{cp}_2, \mathbf{n}_2), \tag{7.1}$$

where $\mathbf{n}_1$ and $\mathbf{n}_2$ are specified by the definition of l_n. In this work, we only consider the case that $\mathbf{n}_1$ and $\mathbf{n}_2$ are not parallel. In other words, l_{n1} intersects with l_{n2}. If the friction coefficient is known, $l_{fl1}, l_{fr1}, l_{fl2}.l_{fr2}$ can be also specified.

7.1.2.3 Moment Labels for Two-Edge-Contact Configuration

Moment labeling gives a graphical representation of the composite wrench cone. Each contact has a friction cone associated with the contact point whose boundary

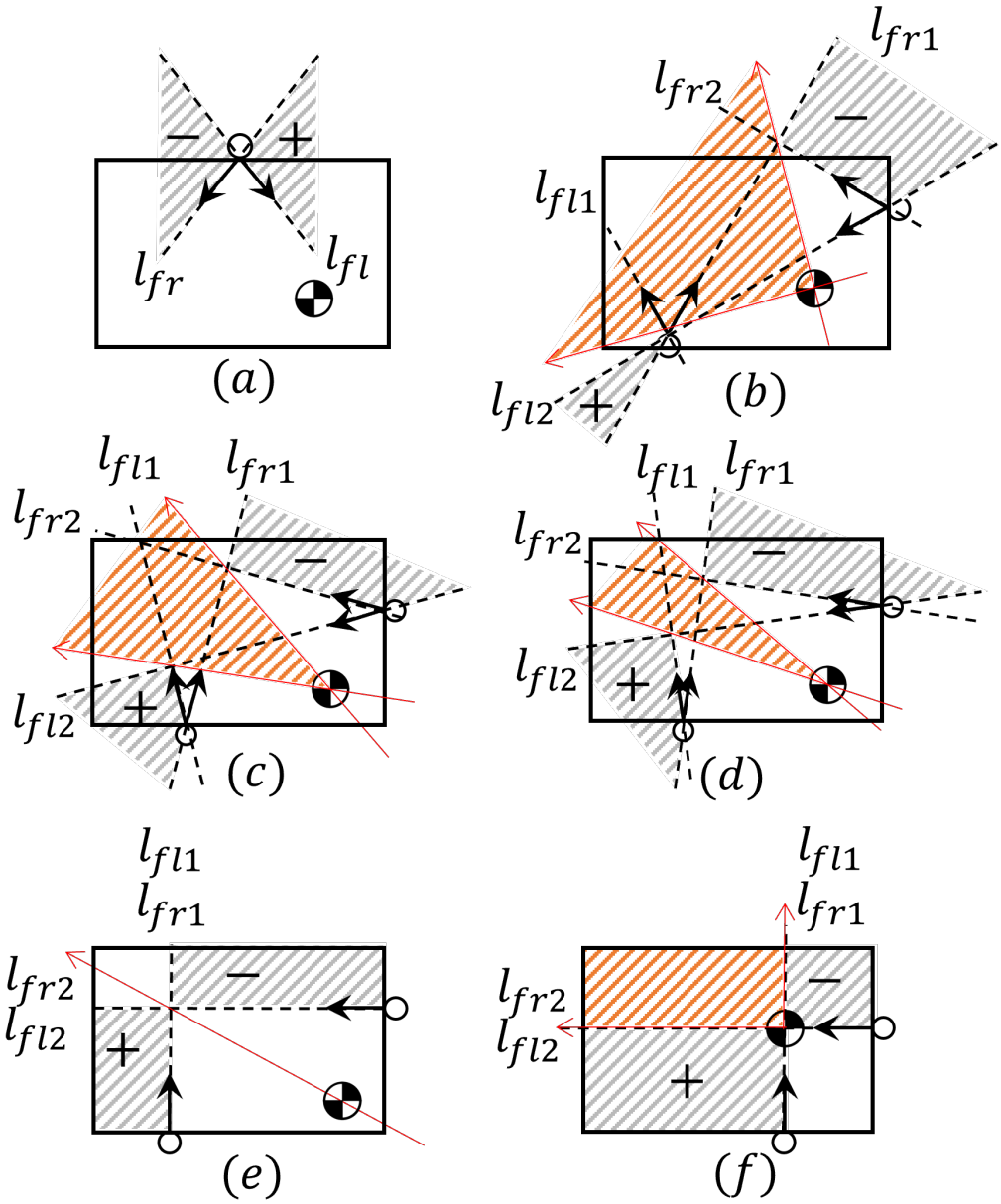


Figure 7.2: Moment labeling representation of pusher-slider system. As in Fig. 7.1, pushers are denoted by small circles which make contact with the slider represented by a rectangle. The CoM has an offset from the centroid. Black arrows delimit the friction cone at the contact. Red regions show the pushing directions along which the object can be translated using the contact configuration employed.

is determined by the friction coefficient between the pusher and the object. Fig. 7.2(a) shows a moment labeling of a composite wrench cone for a single-point contact. The gray shaded regions represent the sign of moment that the resultant force would generate about the region. This can be interpreted as all of the forces that have a different sign of moment about the shaded region or the forces that pass through the shaded region cannot be generated.

Figs. 7.2(b)-(f) show moment labeling representations for two-edge-contact configurations. The moment labels for each contact can be combined to represent the composite wrench cone. The red shaded regions show the composite wrench cone passing through the CoM with two-edge-contact configuration, where any resultant wrench inside the composite wrench cone satisfies the force and moment conditions. Therefore, the composite wrench cone also specifies the directions along which the object can be translated.

Figs. 7.2(b)-(e) show the same $\mathbf{C}_{tec}$ with gradually reduced friction cones, and the red shaded regions become narrower correspondingly. Fig. 7.2(e) is a special case that the friction coefficient between the pusher and the object is equal to zero. In this case, shaded regions converge to a single point. The resultant forces must pass through this intersection point. Therefore, all of the resultant forces that also pass through the CoM have the same direction, which is from the CoM to the intersection of the shaded regions. This set of forces is always a subset of the set of forces generated in Fig. 7.2(b)-(d). In other words, this direction can be always achieved by the contact configuration regardless of friction cones. When two contact normal forces pass through the CoM as shown in Fig. 7.2(f), the resultant forces are positively spanned by these two normal forces even though there is no friction between the pusher and the object.

Theorem 1 *Given l_d and a pushing direction, if there exists a two-edge-contact configuration with which the corresponding contact normal forces positively span*

the pushing direction, also l_{n1}, l_{n2} and l_d intersect at a single point, then the object can be translated along the pushing direction using the two-edge-contact configuration.

Based on the moment labeling analysis of Fig. 7.2(e), given a known object with $\mathbf{C}_{tec}$, we can always find a pushing direction along which the object is translated.

7.1.2.4 Contact Position Tolerance Analysis

Robots may not precisely achieve a desired contact configuration due to various types of image noise. Therefore, it is of importance to investigate under what circumstances the contact noise can be tolerated. As shown in Fig. 7.3, given a known object and its pushing direction, we can draw l_d. For a $\mathbf{C}_{tec}$, the set of resultant wrenches passing through the CoM can be illustrated by the red region. A $\mathbf{C}_{tec}$ is valid, if l_d lies inside the composite wrench cone. Let us examine two-edge-contact configurations by the orange and green pusher in Fig. 7.3(a)-(b), where the blue arrow is the pushing direction and l_d is coincident with the boundary of the wrench cone. Keeping the position of the orange pusher on the horizontal edge fixed, we change the position of the green pusher. Fig. 7.3(a) shows the lower limit position of the green pusher found by drawing a line that passes through $l_d \cap l_{fr1}$ parallel to l_{fr2}. Similarly, Fig. 7.3(b) shows the upper limit position of the green pusher found by drawing a line that passes through $l_d \cap l_{fl1}$ parallel to l_{fl2}. If we draw a line segment whose endpoints are the lower and upper limit positions of the green pusher, we can obtain the tolerance range of the green pusher position. Similarly, keeping the position of the green pusher fixed, we can find the tolerance range of the orange pusher. In some cases (see Fig. 6(e)), the endpoint may be out of the object boundary. In such cases, the line segment ends on the boundary.

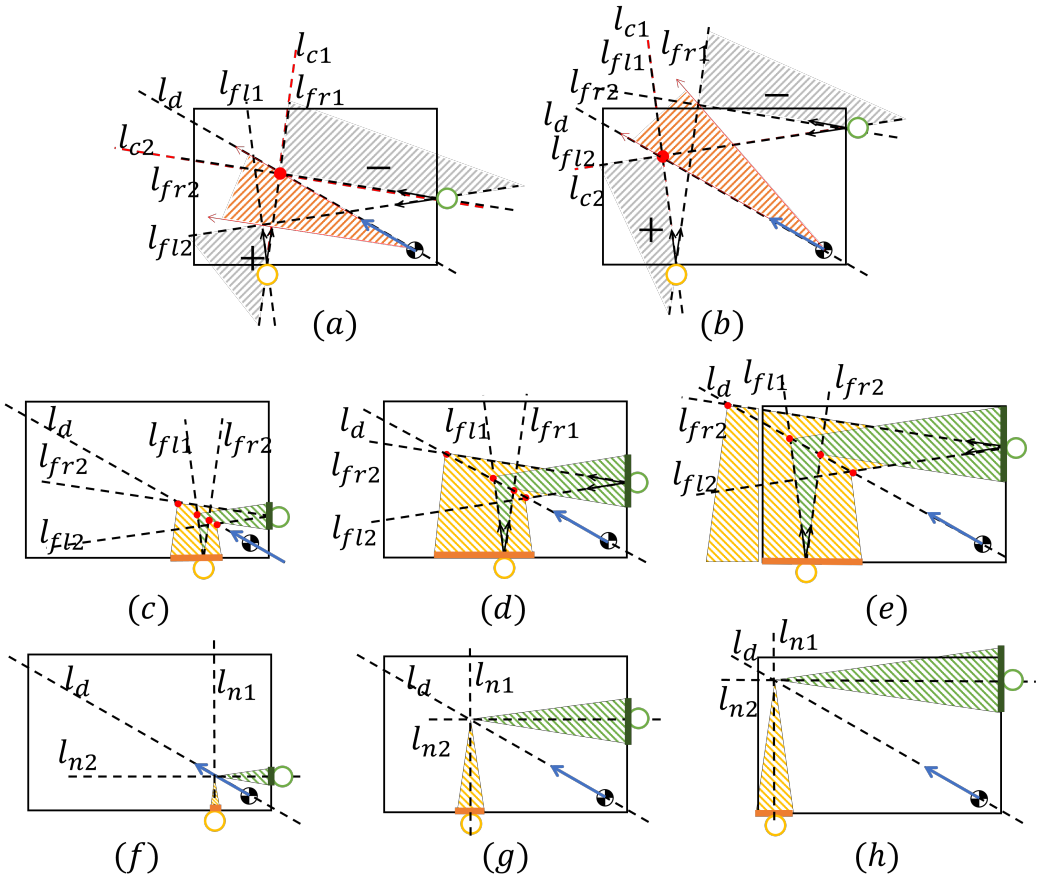


Figure 7.3: Contact position tolerance range. (a) and (b) show the position limits of green pusher when orange pusher remains fixed. (c)-(e) show the tolerance range of each pusher position when the other pusher remains fixed. (f)-(h) shows common tolerance ranges of both pushers' positions within which the object can be purely translated.

Fig. 7.3(c)-(e) show the contact position tolerance ranges represented by colored line segments. We found that, as the distance between the contact points increases, the tolerance range also increases up to a certain point and then decreases as the contact point approaches the object boundary. Also, the tolerance range increases if the friction cone becomes larger.

In practice, the positions of both pushers may be perturbed at the same time. Fig. 7.3(f)-(h) show the common tolerance range of two pushers. In other words, if two pushers select the contact position within the common tolerance range, the net force can be generated for translating the object. This common tolerance range is specified by the position limits of both pushers found as follows: first, we assume that both pushers make contact with the object by Theorem1, and l_d, l_{n1}, and l_{n2} intersect at a single point. Then, we let the orange pusher move toward left (or right) on its edge until l_{fr1} (or l_{fl1}) passes through the intersection point $l_d \cap l_{n2}$. The limits for the green pusher position can be found in a similar way. The common tolerance range depends on the distance between the contact points in the same way as Fig. 7.3(c) to (e).

7.1.2.5 CoM Estimates Tolerance Analysis

Error-prone CoM estimates also cause incorrect contact configuration. Therefore, we need to analyze the tolerance range of the CoM. Given an object with the estimated CoM and a pushing direction, we can draw $\tilde{l_d}$. If the ground truth CoM is on $\tilde{l_d}$, there is no estimation error introduced, because we select $\mathbf{C}_{tec}$ based on Theorem 1. Fig. 7.4 shows four two-edge-contact configurations for pushing an object along the blue arrow with an estimated CoM. Based on the friction cone, we can obtain the CoM tolerance range by the following procedure:

- Find the intersection points $l_{fl1} \cap l_{fl2}$ and $l_{fr1} \cap f_{fr2}$.
- Draw two boundary lines passing through one intersection point parallel to

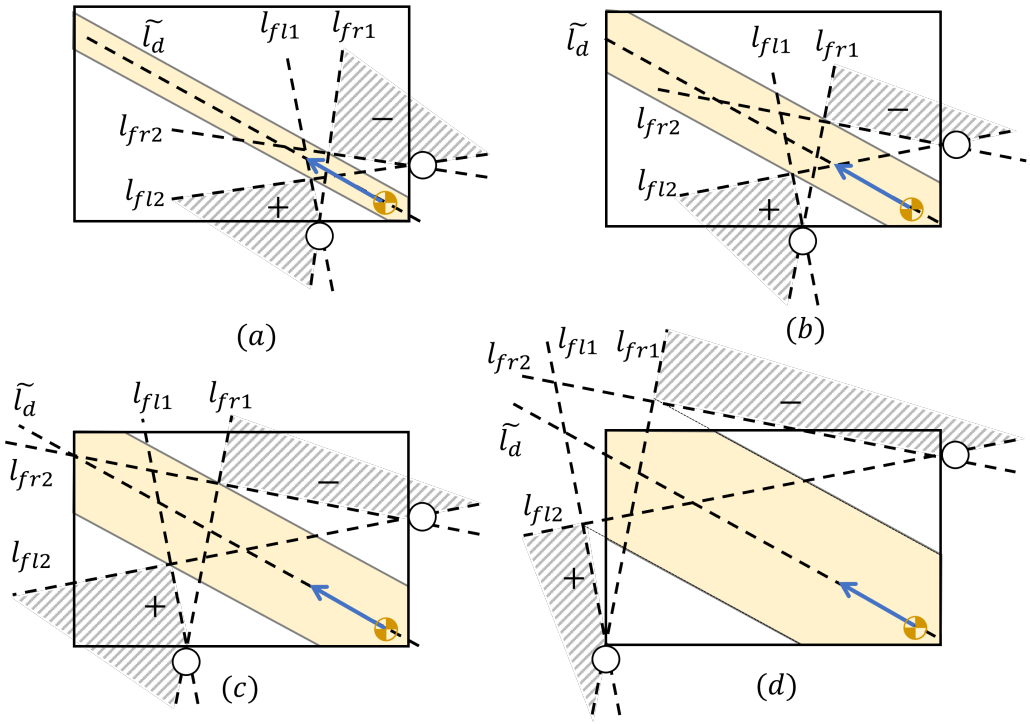


Figure 7.4: CoM tolerance range given the two-edge-contact configurations.

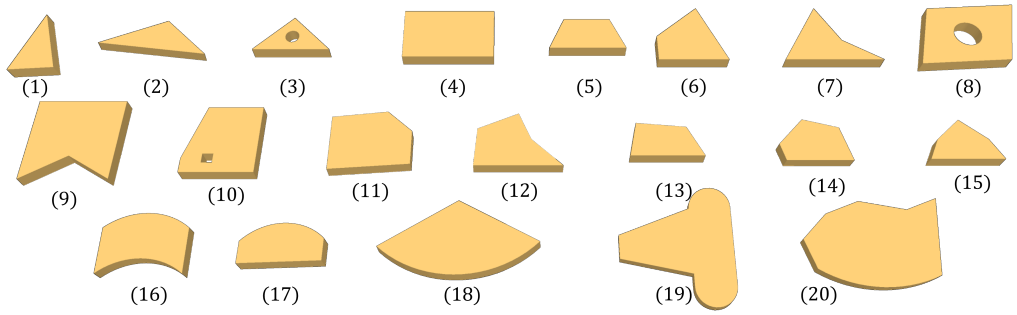


Figure 7.5: The twenty object shapes used in the simulation experiments.

the pushing direction.

- The CoM tolerance range is wrapped by the boundary lines and the object boundary.

If the ground truth CoM is inside the yellow shaded region, the estimated CoM can be tolerated. In other words, the yellow shaded region denotes the tolerance range of the CoM estimates. With the distance between the pushers becomes larger, the tolerance range enlarges. The tolerance range will also be larger for the same two-edge-contact configuration if the friction cone becomes larger. It can be observed that the two-edge-contact configuration with a large distance between two contact points shows high tolerance toward inaccurate estimates.

We now propose our ZMTEP method in order to find two-edge-contact configurations for pushing a novel object as shown in Algorithm 7. Given a number of sampled contour points with associated contact normals, the CoM position, and the pushing direction, this algorithm outputs the most suitable two-edge-contact configuration in which the distance between two contact points is maximized. As the tolerance range becomes smaller due to near-corner contact selection, we remove the contour points near object corners. When dealing with a novel object, ZMTEP estimates the CoM using the method proposed in Section 5.1.

Algorithm 7: Zero Moment Two-Edge Pushing

Input: $\{\mathbf{cp}_i\}^m, \{\mathbf{n}_i\}^m, \mathbf{P}_{CoM}, d_p$

/* $\{\mathbf{cp}_i\}^m$, $\{\mathbf{n}_i\}^m$ are the sampled contour points and the associated normal direction. $\mathbf{P}_{CoM}$ is the position in image frame. d_p is the pushing direction. */

Output: $\mathbf{C}^*_{tec}$

/* $\mathbf{C}^*_{tec}$ refers to a two-edge contact configuration */

$\mathbf{C}^*_{tec} \leftarrow \emptyset$ // initialize $\mathbf{C}^*_{tec}$

$D \leftarrow 0$ // initialize a variable representing the distance between two contact points

for $\mathbf{cp}_i$, $\mathbf{n}_i$ *in* $\{\mathbf{cp}_i\}^m, \{\mathbf{n}_i\}^m$, *i is from 1 to m* **do**

  Get l_{n1} based on $\mathbf{cp}_i$ and $\mathbf{n}_i$

  Get l_d based on $\mathbf{P}_{CoM}$ and d_p

  Get $l_{n1} \cap l_d$

  for $\mathbf{cp}_j$, $\mathbf{n}_j$ *in* $\{\mathbf{cp}_j\}^m$, $\{\mathbf{n}_j\}^m$, *j is from i to m* **do**

    Get l_{c2} based on $\mathbf{cp}_j$ and $\mathbf{n}_j$

    /* $\|a,b\|$ represents the Euclidian distance between a and b. ε is a small value */

    if $\|l_{n2} \cap d_p, l_{n1} \cap l_d\| < \varepsilon$ **then**

      if $\|\mathbf{cp}_i, \mathbf{cp}_j\| > D$ **then**

        $\mathbf{C}^*_{tec} \leftarrow (\mathbf{cp}_i, \mathbf{n}_i, \mathbf{cp}_j, \mathbf{n}_j)$

        $D \leftarrow \|\mathbf{cp}_i, \mathbf{cp}_j\|$

return $\mathbf{C}^*_{tec}$

7.2 Object Pushing Experiments

7.2.1 Simulation Experiment

We conduct three simulation experiments to

- find feasible sets of two-edge contact configuration.
- evaluate the effect of the surface and pusher friction.
- evaluate the performance of ZMTEP for different objects.

In the first simulation, we find the feasible two-edge-contact configurations that translate the object (11) in Fig. 7.5 along a pushing direction. Here both μ_s and μ_p are set to 0.1. The two contact points are selected on different edges whose normal directions are not parallel. There are ten different choices to select two edges. We uniformly sample the contact points from each of the two edges and combine each pair of contact points as a two-edge-contact configuration. A push is considered a success if the object is translated to a goal pose within $0.5cm$ and 5°, respectively.

We show four examples in Fig. 7.6, where each two-edge-contact configuration $\mathbf{C}_{tec}$ forms an intersection point $l_{n1} \cap l_n 2$. We observed that almost all of the feasible two-edge-contact configurations represented by the blue circle marks are found by combining the edges whose contact normals can positively span the pushing direction. In this figure, the blue intersection points are distributed symmetrically along the blue arrow passing through CoM aligned in the pushing direction denoted as l_d in Section 7.1.2.1. The closer the intersection point to l_d, the less the change in object orientation during pushing. The two-edge-contact configurations whose intersection points are on the blue arrow satisfy Theorem1 and therefore result in pure translational motion. Also, the number of blue circle marks increased in the direction of l_d arrowhead including those marks located away from l_d, leading to an increase in the number of suitable configurations available. This phenomenon can be explained by the CoM estimates tolerance analysis. For those contact configurations whose normal directions positively span the pushing direction, each contact configuration obeys Theorem1 but the CoM estimates may not be accurate. When moving closer to the arrowhead, the

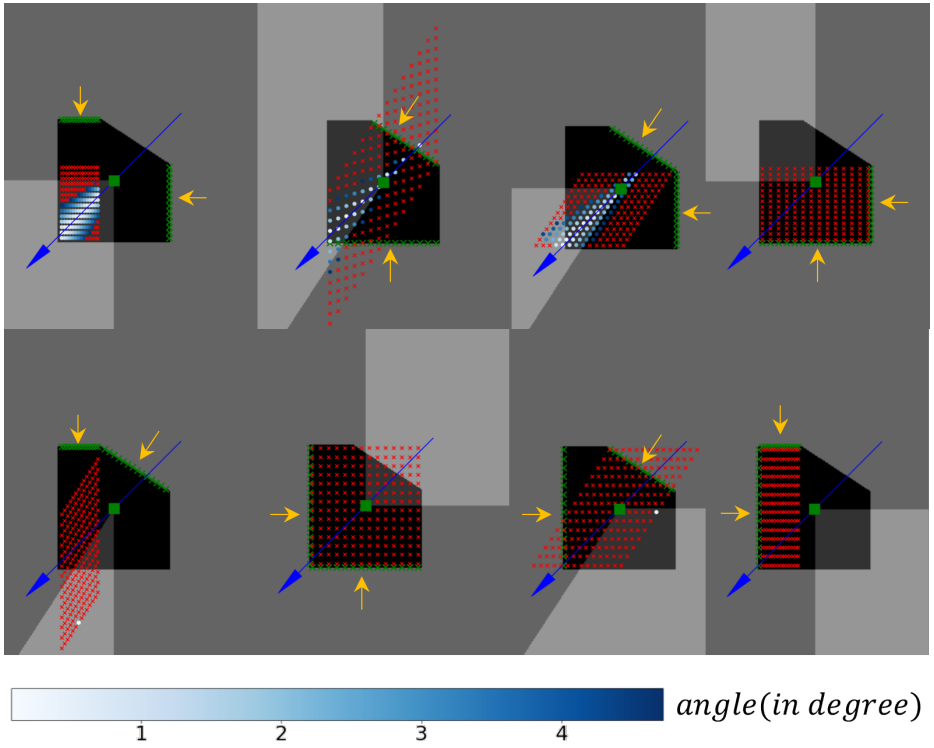


Figure 7.6: Results of object translation simulation. Blue arrow shows the pushing direction and green square mark shows the CoM. Green cross marks on object edges represent the sampled contact points. Yellow arrows show the contact normals that can positively span bright gray regions. A pair of contact points are selected from the sampled contact points on each of two edges. Red and blue cross marks are intersection points formed by two-edge-contact configurations. Red marks are fail and blue marks are success.

distances between two pushers increases so that the configuration shows high tolerance to the error of CoM estimates. Observing the distribution of the red cross marks, the failure can be a result of the following three reasons: (1) the corresponding normal directions cannot positively span the pushing direction, such as the second row of Fig. 7.6 (2) the two-edge-contact configuration whose corresponding intersection point are too far away from l_d, and (3) the distance between the contact points is too narrow so that even a small error in contact points results in failure.

In the second simulation, we use nine different friction settings by permuting three different friction coefficients $(0.1, 0.5, 1.0)$. In each of the settings, we push the triangular object (1) shown in Fig. 7.5 along random directions. We select two edges whose normal directions can positively span the pushing direction. We then combine each contact point on each edge to find feasible two-edge push configurations. A push is considered a success using the same criterion as the first simulation experiment. From the result of the second simulation experiment, we found that with the increase of the pusher friction or surface friction, the number of successful pushing has also increased. We conclude that both the friction between the pusher and the object and the friction between the surface and the object mutually contribute to enlarging the contact tolerance range. Similar to the first pushing simulation experiment, the number of intersection points increased in the direction of l_d arrowhead for each of the frictional setting, which leads to the same conclusion that the contact configuration having larger distance between two contact points exhibits high CoM estimates tolerance. In addition, as each row or column of intersection points along the contact normals can be seen as a set of contact configurations that one pusher position gets perturbed when the other remains fixed. Along the pushing direction, when counting the number of success pushes in rows or columns, the number of success pushes increases up

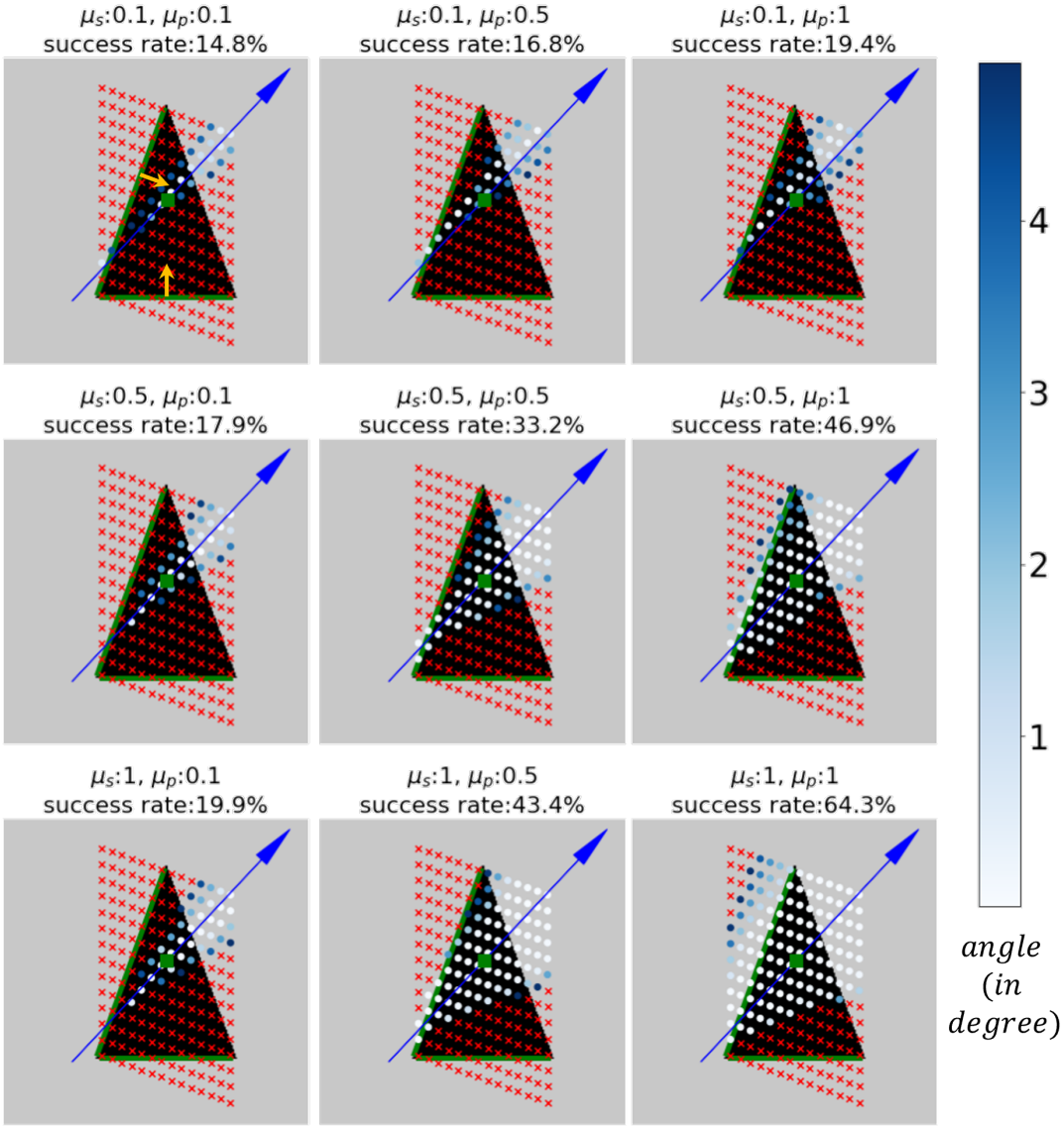


Figure 7.7: Nine simulation environments with different surface and pusher frictions. Success rate is computed by calculating the ratio between the total number of success pushes and the total number of pushes.

Table 7.1: Object Translation Simulation Experiment

Object	**Low Friction**				**High Friction**			
	Proposed1	**Proposed2**	**Baseline1**	**Baseline2**	**Proposed1**	**Proposed2**	**Baseline1**	**Baseline2**
1	**1.000**	0.985	0.277	0.362	**1.000**	**1.000**	0.823	0.769
2	**0.942**	0.925	0.050	0.175	**1.000**	**1.000**	0.350	0.500
3	**1.000**	0.975	0.117	0.242	**1.000**	0.983	0.608	0.567
4	**0.975**	0.900	0.267	0.492	**1.000**	**1.000**	0.992	0.733
5	**0.984**	0.952	0.234	0.419	**0.960**	**0.960**	0.863	0.677
6	**1.000**	0.921	0.414	0.443	0.979	**0.993**	0.957	0.786
7	**0.938**	0.906	0.312	0.328	**1.000**	0.969	0.744	0.721
8	**0.981**	0.915	0.321	0.481	**1.000**	**1.000**	0.849	0.660
9	**0.905**	0.781	0.248	0.489	**0.978**	0.942	0.964	0.761
10	**0.985**	0.916	0.298	0.511	**1.000**	**1.000**	0.955	0.674
11	**0.992**	0.961	0.403	0.597	0.992	**1.000**	0.891	0.845
12	**0.949**	0.934	0.301	0.368	**1.000**	0.993	0.853	0.757
13	**0.967**	0.919	0.366	0.528	0.992	0.992	**1.000**	0.815
14	**1.000**	0.835	0.339	0.543	**1.000**	**1.000**	0.808	0.752
15	**1.000**	0.976	0.317	0.357	**1.000**	**1.000**	0.762	0.683

to a certain point and then deceases, which is supported by the contact position tolerance analysis.

In the third simulation, we verify the ZMTEP using the objects in the first two rows in Fig. 7.5 in a low friction setting (μ_s and μ_p are 0.1) and a high friction setting (μ_s and μ_p are 1.0). Each object assigned to ten different CoMs is translated $25cm$ along the fifteen uniformly sampled directions. Henceforth, a push is considered a success if the slider is within $1cm$ and 10° of a goal pose.

For comparison purpose, we used following methods:

- **Proposed1** selects two contact points by ZMTEP, given the CoM.
- **Proposed2** selects two contact points by ZMTEP, estimating the CoM.
- **Baseline1** uses two contact points that have the maximum distance from the slider's centroid along the direction perpendicular to the pushing direction. Increasing the distance between two contact points, errors of contact location and CoM estimation can be better tolerated.
- **Baseline2** uses two contact points that are the largest possible equal distance away from the given CoM along the direction perpendicular to the pushing direction. This method relies on the CoM ground truth and ensures that the CoM is in the middle between two contact points along the pushing direction. The distance between two contact points tends to be smaller than **Baseline1**.

The obtained results in the form of mean ratio of successful pushes are presented in Table 7.1. From the result, in a low friction environment, **Proposed1** performed best and **Proposed2** is the second best. The baseline methods performed far less accurately than our proposed methods. **Baseline1** performs worse than **Baseline2**, as the tolerance capability is restricted by the limited friction. **Baseline2** reaps the benefits of knowing the CoM ground truth. In high friction environment, the performance of **Proposed1** and **Proposed2** improved further.

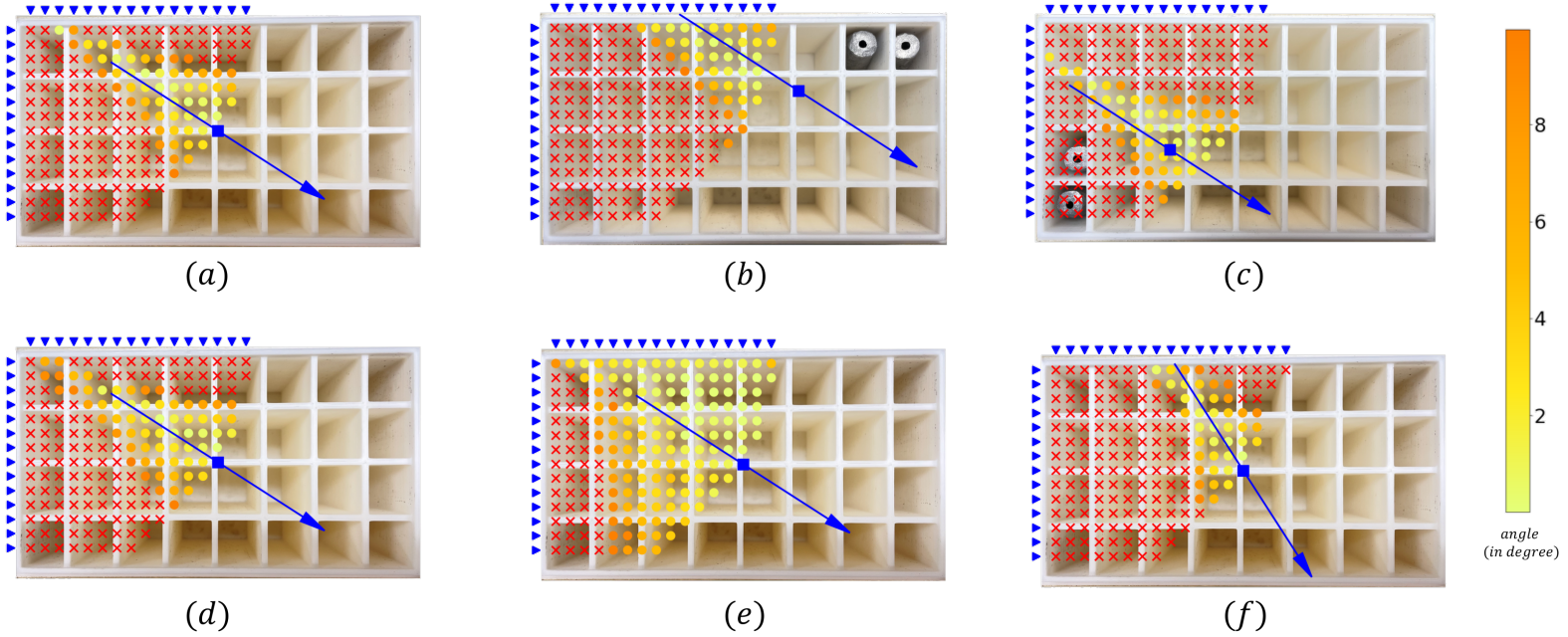


Figure 7.8: Real robot pushing of a box with a variable stroke gripper along the blue arrow direction. Blue triangle markers are sampled contact points. Each pair of contact points form an intersection point denoted by red cross markers and orange dot markers. Blue squares are the CoMs of the grid box. The CoMs in (b) and (c) are biased by two lead blocks near the corner.

Compared with the baseline methods in a low friction environment, even though the two-edge contact configurations in high friction setting are the same as the ones in low friction setting, their performances are greatly improved. **Baseline1** performed better than **Baseline2** as the tolerance capability gets enlarged due to the increase in friction. This result shows that high friction helps expand the wrench cone. Therefore, the performance of baseline methods highly rely on the friction. On the other hand, the proposed methods work well irrespective of frictional conditions. Compared with **Proposed1**, even though **Proposed2** use the estimated CoM which may introduces undesired contact configuration, the performance of **Proposed2** drops only slightly. The reason is that we select a two-edge-contact configuration with a tolerance range.

7.2.2 Real Experiment

Changing the friction properties of the pusher and the plane as well as the CoM of a grid box slider, we find a set of feasible two-edge-contact configurations. The robot uses its parallel-jaw gripper to translate the box slider $25cm$ along a desired direction. Given a pushing direction, we sample 14 contact points on the short edge and 16 contact points on the long edge, yielding a set of 224 paired configurations. As we set the gripper stroke limit to $8.1cm$, some of the paired configurations cannot be achieved, leading to 183 pushes available.

We conduct a total of six experiments as illustrated in Fig. 7.8(a)-(f), where the robot pushes the box in the direction of the blue arrow. In (b) and (c), the position of CoM is biased arising from the addition of lead blocks. In all experiments, the object is placed on a plastic plane pushed by the kraft paper wrapped gripper except (d) and (e). (d) uses a high friction foam surface and (e) uses a rubber wrapped gripper.

Chapter 8

Conclusion

Following the recent works in [13, 14], 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. 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 par with the results in the 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 the 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.

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 effects 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 the prediction model challenging since the dynamics of contact tends to become more complicated.

We presented a novel CoM estimation method by combining Mason's voting theorem and a deep learning motion prediction model trained on our simulation dataset **SimPush**. We demonstrated the process of how the probable CoM location can be narrowed down with no *a priori* assumption about friction between the pusher and object.

We addressed the problem of planar multi-contact pushing of novel objects undergoing pure translational motion. We proposed the two-edge-contact pushing interaction called ZMTEP to translate an object to a goal pose. Using moment labeling representation, we showed that the pushing configuration stated in The-

orem 1 satisfies the condition of pure translation. Thirdly, toward real-world applications suffering from image sensor noise and error-prone CoM estimates, we analyzed the tolerance region of contact configuration to both contact noise and inaccurate CoM estimates. We showed that as the distance between two contact locations increases, the proposed ZMTEP exhibits high tolerance to both issues.

We demonstrated through extensive experiments both in simulation and real robotic pusher-slider settings that the proposed CoM estimation method has good mean squared error properties and small standard deviation and ZMTEP outperformed the baseline methods. To the best of our knowledge, this research is the first attempt to provide a thorough analysis and empirical evidence of multi-edge-contact pushing based on fewer assumptions to achieve a reasonably close optimal shortest path of translational motion. One of the future directions would be investigating multi-edge-contact configuration selection for re-orienting an object before and after an action executes. In addition, pushing an object at a high speed, beyond the quasi-static approximation, would be also of importance to further reduce the time needed for translating an object to a goal pose.

References

[1] M. T. Mason, "Progress in nonprehensile manipulation," *The International Journal of Robotics Research*, vol. 18, no. 11, pp. 1129–1141, 1999. [Online]. Available: https://doi.org/10.1177/02783649922067762

[2] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4627–4632.

[3] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections," in *1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 78, 2017, pp. 344–356. [Online]. Available: http://proceedings.mlr.press/v78/frederik-ebert17a.html

[4] Z. Dong, S. Krishnan, S. Dolasia, A. Balakrishna, M. Danielczuk, and K. Goldberg, "Automating planar object singulation by linear pushing with single-point and multi-point contacts," in *IEEE International Conference on Automation Science and Engineering*, 2019, pp. 1429–1436.

[5] L. Chang, J. Smith, and D. Fox, "Interactive singulation of objects from a pile," *IEEE International Conference on Robotics and Automation*, pp. 3875–3882, 2012.

[6] M. Danielczuk, J. Mahler, C. Correa, and K. Goldberg, "Linear push policies to increase grasp access for robot bin picking," in *IEEE International Conference on Automation Science and Engineering*, 2018, pp. 1249–1256.

[7] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *Robotics: Science and Systems*, vol. 1, 2011.

[8] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4238–4245.

[9] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[10] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction part 1. limit surface and moment function," *Wear*, vol. 143, no. 2, pp. 307–330, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0043164891901043

[11] ——, "Planar sliding with dry friction part 2. dynamics of motion," *Wear*, vol. 143, no. 2, pp. 331–352, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0043164891901054

[12] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *Frontiers in Robotics and AI*, vol. 7, p. 8, 2020.

[13] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "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*, 2016, pp. 30–37.

[14] M. Bauza, F. Alet, Y.-C. Lin, T. Lozano-Pérez, L. P. Kaelbling, P. Isola, and A. Rodriguez, "Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgb-d video," *arXiv preprint arXiv:1910.00618*, 2019.

[15] A. Kloss, M. Bauza, J. Wu, J. B. Tenenbaum, A. Rodriguez, and J. Bohg, "Accurate vision-based manipulation through contact reasoning," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 6738–6744.

[16] Z. Xu, W. Yu, A. Herzog, W. Lu, C. Fu, M. Tomizuka, Y. Bai, C. K. Liu, and D. Ho, "Cocoi: Contact-aware online context inference for generalizable non-planar pushing," *arXiv preprint arXiv:2011.11270*, 2020.

[17] K. N. Kumar, I. Essa, S. Ha, and C. K. Liu, "Estimating mass distribution of articulated objects using non-prehensile manipulation," *arXiv preprint arXiv:1907.03964*, 2019.

[18] J. K. Li, W. S. Lee, and D. Hsu, "Push-net: Deep planar pushing for objects with unknown physical properties," in *Robotics: Science and Systems*, 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p24.html

[19] A. Byravan and D. Fox, "Se3-nets: Learning rigid body motion using deep neural networks," *IEEE International Conference on Robotics and Automation*, no. 3, pp. 173–180, 2017.

[20] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3511–3516.

[21] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, jun 2020. [Online]. Available: https://doi.org/10.1145/3386252

[22] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 533–556, 1996. [Online]. Available: https://doi.org/10.1177/027836499601500602

[23] F. R. Hogan and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 755–773, 2020. [Online]. Available: https://doi.org/10.1177/0278364920913938

[24] J. Zhou, M. T. Mason, R. Paolini, and D. Bagnell, "A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation," *The International Journal of Robotics Research*, vol. 37, no. 2-3, pp. 249–265, 2018. [Online]. Available: https://doi.org/10.1177/0278364918755536

[25] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018. [Online]. Available: https://doi.org/10.1177/0278364917710318

[26] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 3406–3413, 2016.

[27] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," *IEEE International Conference on Robotics and Automation*, pp. 2786–2793, 2017.

[28] Z. Xu, Z. He, J. Wu, and S. Song, "Learning 3d dynamic scene representations for robot manipulation," *CoRR*, vol. abs/2011.01968, 2020. [Online]. Available: https://arxiv.org/abs/2011.01968

[29] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," *arXiv preprint arXiv:1605.07157*, 2016.

[30] A. Eitel, N. Hauff, and W. Burgard, "Learning to singulate objects using a push proposal network," *Springer Proceedings in Advanced Robotics*, vol. 10, pp. 405–419, 2020.

[31] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems*, 2016, pp. 5074–5082.

[32] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction part 1. limit surface and moment function," *Wear*, vol. 143, no. 2, pp. 307–330, 1991.

[33] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects," *CoRR*, vol. abs/1710.04102, 2017. [Online]. Available: http://arxiv.org/abs/1710.04102

[34] W. Goo and S. Niekum, "Local nonparametric meta-learning," *arXiv preprint arXiv:2002.03272*, 2020.

[35] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," *arXiv preprint arXiv:1901.05761*, 2019.

[36] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," *International Conference on Learning Representations*, pp. 1–12, 2016.

[37] C. Song and A. Boularias, "A probabilistic model for planar sliding of objects with unknown material properties: Identification and robust planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 5311–5318.

[38] C. Wang, S. Wang, B. Romero, F. Veiga, and E. Adelson, "Swingbot: Learning physical features from in-hand tactile exploration for dynamic swing-up manipulation," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5633–5640, 2020.

[39] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman, "Physics 101: Learning physical object properties from unlabeled videos," *British Machine Vision Conference*, vol. 2016-September, pp. 39.1–39.12, 2016.

[40] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song, "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions," in *Robotics: Science and Systems*, 2019. [Online]. Available: http://www.zhenjiaxu.com/DensePhysNet/

[41] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *Robotics: Science and Systems*, vol. 13, 2017.

[42] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018.

[43] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections." in *CoRL*, 2017, pp. 344–356.

[44] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *European Conference on Computer Vision*. Springer, 2016, pp. 835–851.

[45] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, "Transporter networks: Rearranging the visual world for robotic manipulation," 2021.

[46] Z. Gao, A. Elibol, and N. Y. Chong, "A 2-stage framework for learning to push unknown objects," in *Joint IEEE International Conference on Development and Learning and Epigenetic Robotics*, 2020, pp. 1–7.

[47] Z. Gao, A. Elibol, and N. Y. Chong, "Non-prehensile manipulation learning through self-supervision," in *IEEE International Conference on Robotic Computing*, 2020, pp. 93–99.

[48] C. Lin, M. Grner, P. Ruppel, H. Liang, N. Hendrich, and J. Zhang, "Self-adapting recurrent models for object pushing from learning in simulation," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5304–5310, 2020.

[49] E. Arruda, M. J. Mathew, M. Kopicki, M. Mistry, M. Azad, and J. L. Wyatt, "Uncertainty averse pushing with model predictive path integral control," *IEEE-RAS International Conference on Humanoid Robots*, pp. 497–502, 2017.

[50] J. Wang, C. Hu, Y. Wang, and Y. Zhu, "Dynamics learning with object-centric interaction networks for robot manipulation," *IEEE Access*, vol. 9, pp. 68 277–68 288, 2021.

[51] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani, "Object-centric forward modeling for model predictive control," *arXiv*, no. CoRL, pp. 1–13, 2019.

[52] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," 2019.

[53] N. Mavrakis and R. Stolkin, "Estimation and exploitation of objects ' inertial parameters in robotic grasping and manipulation: A survey," *Robotics and Autonomous Systems*, vol. 124, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889019302313

[54] T. Standley, O. Sener, D. Chen, and S. Savarese, "image2mass: Estimating the mass of an object from its image," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 78, 2017, pp. 324–333. [Online]. Available: https://proceedings.mlr.press/v78/standley17a.html

[55] Y. Yu, T. Arima, and S. Tsujio, "Estimation of object inertia parameters on robot pushing operation," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 1657–1662.

[56] N. Mavrakis, A. M. Ghalamzan E., and R. Stolkin, "Estimating an object's inertial parameters by robotic pushing: A data-driven approach," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 9537–9544, 2020.

[57] K. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 1992, pp. 416–421.

[58] A. Allevato, E. S. Short, M. Pryor, and A. Thomaz, "Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer,"

in *Proceedings of the Conference on Robot Learning*, vol. 100, 2020, pp. 445–455.

[59] A. Allevato, M. Pryor, and A. Thomaz, "Multi-parameter real-world system identification using iterative residual tuning," in *Proceedings of the ASME International Design and Technical Conference*, 2020.

[60] T. Hermans, F. Li, J. M. Rehg, and A. F. Bobick, "Learning contact locations for pushing and orienting unknown objects," in *IEEE-RAS International Conference on Humanoid Robots*, 2013, pp. 435–442.

[61] S. Krivic and J. Piater, "Online adaptation of robot pushing control to object properties," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4614–4621.

[62] C.-Y. Chai, W.-H. Peng, and S.-L. Tsao, "Adaptive unknown object rearrangement using low-cost tabletop robot," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 2372–2378.

[63] Q. Li and S. Payandeh, "Manipulation of convex objects via two-agent point-contact push," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 377–403, 2007. [Online]. Available: https://doi.org/10.1177/0278364907076819

[64] J. Lloyd and N. F. Lepora, "Goal-driven robotic pushing using tactile and proprioceptive feedback," *IEEE Transactions on Robotics*, pp. 1–12, 2021.

[65] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[66] "CM Labs Vortex Studio Academic," https://www.cm-labs.com/vortex-studio/software/vortex-studio-academic-access/, accessed: 2020-09-30.

[67] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects from sensory data," *The International Journal of Robotics Research*, 2020. [Online]. Available: https://doi.org/10.1177/0278364920954896

[68] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[69] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.

[70] Z. Gao, A. Elibol, and N. Y. Chong, "Planar pushing of unknown objects using a large-scale simulation dataset and few-shot learning," in *IEEE International Conference on Automation Science and Engineering*, 2021, pp. 341–347.

[71] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235

[72] M. T. Mason, *Mechanics of Robotic Manipulation*. Cambridge, MA, USA: MIT Press, 2001.

Publications

[1] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "A Few-Shot Learning Framework for Planar Pushing of Unknown Objects," Intelligent Service Robotics (2021 Impact Factor 2.468), https://doi.org/10.1007/s11370-022-00425-7, May 21, 2022

[2] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "Zero Moment Two Edge Pushing of Novel Objects with Center of Mass Estimation," IEEE Transactions on Automation Science and Engineering (2021 Impact Factor 6.636), Conditionally Accepted April 20, 2022.

[3] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "Estimating the Center of Mass of an Unknown Object for Nonprehensile Manipulation," 2022 IEEE International Conference on Mechatronics and Automation, August 7-10, 2022 (Accepted for Oral Presentation).

[4] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "Estimating the Center of Mass of an Unknown Object via Dynamic Pushing," 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), August 23-27, Industry paper, 2022 (Accepted for Oral Presentation)

[5] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "Planar Pushing of Unknown Objects Using a Large-Scale Simulation Dataset and Few-Shot Learning," 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), August 23-27, 2021, pp. 341-347, doi:

10.1109/CASE49439.2021.9551513.

[6] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "A 2-Stage Framework for Learning to Push Unknown Objects," 2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), November 26-27, 2020, pp. 1-7, doi: 10.1109/ICDL-EpiRob48136.2020.9278075.

[7] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "Non-Prehensile Manipulation Learning through Self-Supervision," 2020 Fourth IEEE International Conference on Robotic Computing (IRC), November 9-11, 2020, pp. 93-99, doi: 10.1109/IRC.2020.00022.

[8] Ziyan Gao, Armagan Elibol, and Nak Young Chong, "A Hybrid 2-Stage Method for Robotic Planar Pushing," Proceedings of the International Conference on Ubiquitous Robots (UR), Kyoto, Japan, June 22-26, 2020 (Work-in-Progress Presentation).

[9] Ziyan Gao and Nak Young Chong, "Efficient Robotic Grasp Learning by Demonstration," Proceedings of International Conference on Robot Intelligence Technology and Applications; Springer Lecture Notes in Mechanical Engineering, pp. 87-99, Putrajaya, Malaysia, Dec. 16-18, 2018.