Estimating the Center of Mass of an Object with Non-uniform Density via High-speed Pushing

Ziyan Gao, Armagan Elibol, and Nak Young Chong

***Abstract*— An object's inertial parameters, such as the mass, the Center of Mass (CoM), and the moment of inertia, affect the response to the external forces exerted on it. It is important to estimate these parameters as accurately as possible in order to facilitate robot-led automation including grasping and manipulation. Traditionally, the estimation is conducted by employing special equipment in a controlled environment, which may not be always available for a small batch production system dealing with unknown objects. We propose an efficient exploratory type framework for estimating an object's CoM via force sensor-less high-speed robotic pushing, which only requires the use of a vision system for detecting the change in the object's pose. Accurately estimating an object's CoM aids robotic grasping and expands manipulation scenarios. We conducted intensive simulation and real robot experiments to show the accuracy of the estimation, robustness to friction variation, and generalization capability to novel objects, that the framework ensures only with a limited number of pushes.**

I. INTRODUCTION

The inertial parameters of an object affect its response to robot actions performed on the object. These parameters are usually obtained using special equipment in a controlled environment, which may not be suitable for industrial settings. Accurate identification of object inertial parameters is essential to infer efficient robot grasping and manipulation [1]. Therefore, a simple yet accurate method for inertial parameter identification with minimal equipment is of wide interest to the robotic manipulation community. Among the inertial parameters, we focus on the CoM as it helps understand the object's behavior being manipulated. For instance, the robot needs to slide a thin object on a table beyond the table edge to pinch grasp it. The motion of the robot should ensure that the object CoM always lies inside the table top to prevent it from falling to the floor. When pushing a planar object on a uniform surface, the CoM helps infer the object's sense of rotation given the frictional contact forces [2]. Likewise, our previous work [3] has shown that the robot can robustly translate unknown planar objects if their CoMs are well estimated.

In this work, we aim to estimate the CoM of novel objects via planar pushing using only a position-controlled robot arm and an off-the-shelf vision system. The projection of object CoM to the horizontal plane will have the same location as the center of friction if all the supported contacts have the same frictional coefficient. Rooted back to the work by Lynch [4] and our previous works [3][5], the object center of friction can be estimated by choosing a set of test pushes to constrain the center of friction to a convex region (hereinafter referred to as the CoM region) inside the convex hull of the object following Mason's Voting Theorem (VT) [2]. We propose a new framework based on a recurrent neural network (RNN) trained by a simulation dataset and VT [2] to acquire an accurate estimation on the object's CoM, taking the advantages from the RNN and CoM region. The estimation is conducted using a sequence of pushing interactions. To minimize the number of pushing interactions and improve the performance of RNN, we propose an evaluation function to select a sequence of pushing actions considering how much the CoM region can be narrowed down and how the region can be shaped. The simulation and real experimental results show the significant generalization capability and robustness of the framework.

In our previous work [3][5], we estimated the object's CoM through a motion prediction model which relies on a set of pushing priors, and only considered quasi-static pushing. In this work, without using pushing priors for the prediction of motion, the proposed framework can accurately estimate the object's CoM with fewer pushes. We conduct a series of simulation and real robot experiments to compare the impact of pushing speed and floor surface friction on the estimation accuracy and evaluate the proposed framework in real settings. We found that both higher speed pushing and lower frictional settings contribute to improving the estimation accuracy. In summary, this letter introduces the following contributions:

- A novel object CoM estimation method only uses a position-controlled robotic pusher and vision system.
- The estimation framework incorporating the voting theorem and deep learning model improves the estimation efficiency and accuracy.
- Quantitative evaluation of the effect of pushing speed and friction on the estimation accuracy.

The assumptions are made as follows:

- The pushed object is flat, and it does not tilt or flip during and after being pushed.
- The pusher, the object, and the support plane are rigid.
- Coulomb's law of friction applies.
- The contact normal is extracted from the vision system.

II. RELATED WORK

Following the taxonomy in [1], inertial parameter estimation methods can be classified as purely visual methods, fixed-object methods, and exploratory methods. The purely visual methods employ vision sensors to measure the volume

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

of an object, assuming uniform mass density. Leveraging the large-scale labeled dataset, Trevor *et al.* [6] proposed a deep learning model to estimate both the volume and density of the object using an RGB-D image. In the fixed-object methods [7], the object is firmly attached to the robot's end-effector, and the rigid body dynamics is analyzed with object shape information. The focus of this work lies on the exploratory methods that require the robot to interact with objects, as detailed below.

Yu *et al.* [8] used a two-finger pushing in a trial-and-error approach making the line of CoM pass between the fingers. This method might be inefficient when dealing with an arbitrarily shaped object. Tsuneo *et al.* [9] uniformly sampled a set of hypothesized support points based on the object shape obtained from the vision sensor, and the support frictional forces were solved by unconstrained least-squares. However, this method may result in unrealistic friction distribution as reported in [4]. On the other hand, Lynch [4] formulated the estimation of the center of friction as a linear programming problem, and a set of support points were hypothesized. However, the number of data points was required to be much larger than the number of hypothesized support points in order to obtain an accurate estimation. On the other hand, with an ellipsoid force-model model presented in [10], Kloss *et al.* [11] used an Extended Kalman Filter to iteratively estimate object inertial parameters based on the applied pushing action and resultant object motion. In addition, Song *et al.* [12] proposed to learn the coupled mass-friction parameters using a differentiable simulator, and obtain the friction distribution minimizing the simulation-reality gap. This method requires a set of hypothesized mass and friction models and the object coarsely approximated by rigidly-connected 2D small grids.

Recently, learning-based methods have been employed for estimating inertial parameters. McGovern *et al.* [13] utilize reinforcement learning for CoM estimation by placing the object to the border of the table to check whether the object is balanced or not. However, in a real setting, this method lacks the efficiency and stability as the robot should precisely pick and place the object to the desired pose. Li *et al.* [14] used RNNs to sample actions for pushing objects toward the desired pose, and the object CoM is predicted on the fly taking into account the interaction history as input. Kumar *et al.* [15] employed a policy network to interact with an articulated object, and a predictor network to predict the mass distribution of the object. Different from these works, we sample pushing actions based on the CoM region in such a way as to reduce the CoM uncertainty as much as possible. Xu *et al.* [16] proposed a learning framework to encode the object's physical properties implicitly using high-speed pushing and colliding. It is worth mentioning that pushing the object at high speed makes the object's physical properties more distinguishable than pushing at low speed. This is consistent with our experimental results. Incorporating the learning model and physics engine, Allevato *et al.* [17], [18] 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 using a physics engine, Wu *et al.* [19] fed explicitly estimated physical parameters as input to an analytical model of a physical system to estimate object motion. Veres *et al.* [20] proposed to learn the CoM implicitly and predict the grasp affordance in an end-to-end fashion. Specifically, several grasping trials were executed to collect the support set for generating the grasp affordance.

In this work, we aim to improve the estimation efficiency, leveraging the CoM region and RNN, and obtain more accurate CoM estimates using high speed pushing under varying conditions of surface friction.

NOMENCLATURE

$\mathbf{M}_{CoM}$	CoM region in matrix form comprising all pixel coordinates within it. It has m rows and two columns, where m specifies the number of pixel coordinates. $\mathbf{M}_{CoM_j}$ is a pixel coordinate at the jth row.
t	The leading superscript t refers to the time step.
$\mathbf{P}_{ct}$	Matrix representation with n rows and 2 columns of all sampled pixel coordinates on the object outline, where n specifies the number of sampled contact positions. $\mathbf{P}_{ct_j}$ is a pixel coordinate at the jth row.
$\mathbf{N}_{ct}$	Matrix representation of all sampled normal directions associated with $\mathbf{P}_{ct}$. $\mathbf{N}_{ct}$ has the same dimension as $\mathbf{P}_{ct}$. $\mathbf{N}_{ct_j}$ is an unit vector indicating the normal direction at the jth contact position calculated based on the adjacent pixels at $\mathbf{P}_{ct_j}$.
$\mathbf{p}_{CoM}$	CoM ground truth. $\hat{\mathbf{p}}_{CoM}$ represents CoM estimates.
$\mathbf{p}_{cntr}$	Centroid of the CoM region.
$\hat{\mathbf{p}}_{rnn}$	CoM estimates using RNN.
$\mathbf{a}$	A 4-dimensional vector specifying a pushing direction that consists of $\mathbf{P}_{ct_j}$ and $\mathbf{N}_{ct_j}$.
$\Delta\mathbf{o}$	A 3-dimensional vector specifying the object's translation (Δx, Δy) and rotation ($\Delta\theta$) in a plane.
C_{bw}	a threshold distance between a pixel and the pushing direction.
θ_T	a threshold angle of the object's rotation in radians.

III. METHOD

We introduce a new framework for estimating an object's CoM that requires a robot to interact with the object, as illustrated in Fig. 1. This framework estimates the object's CoM iteratively until the uncertainty measured by the CoM region is smaller than a prespecified area threshold or the number of interactions reaches the maximum allowed limit. The framework consists of five modules, among which is the CoM region updater playing a core role in both the robot-object interaction loop and the CoM estimation phase. In the interaction loop, the CoM region updater passes the CoM region to the contact selector in order to guide the action selection procedure. In the CoM estimation phase, the CoM region updater updates the CoM region, allowing the combine module to examine the spatial relation between the CoM region and the output of RNN and further improves the estimation accuracy. We will introduce each module in the following parts.

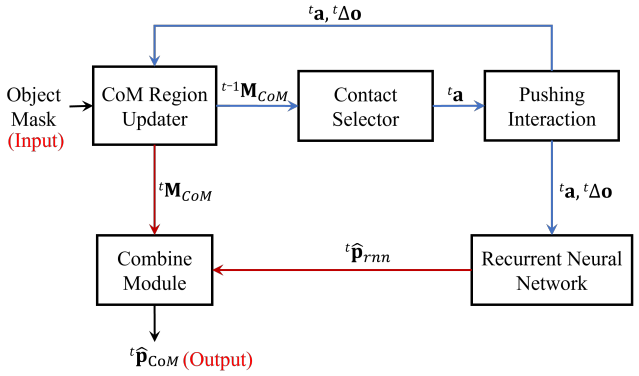

Fig. 1. Proposed framework for CoM estimation. The CoM region updater takes the object mask as input to construct the CoM region ${}^{t-1}\mathbf{M}_{CoM}$. In the pushing interaction loop shown in blue, the CoM updater passes ${}^{t-1}\mathbf{M}_{CoM}$ to the contact selector to sample a pushing action ${}^{t}\mathbf{a}$. The robot-object interaction module exerts a push on the object using ${}^{t}\mathbf{a}$ and observes the pose change ${}^{t}\Delta\mathbf{o}$. In the CoM estimation phase shown in red, employing ${}^{t}\mathbf{a}$ and ${}^{t}\Delta\mathbf{o}$, the CoM region updater updates the CoM region ${}^{t}\mathbf{M}_{CoM}$, and the RNN predicts the CoM ${}^{t}\hat{\mathbf{P}}_{rnn}$. The combine module produces a compromise between ${}^{t}\mathbf{M}_{CoM}$ and ${}^{t}\hat{\mathbf{P}}_{rnn}$ for the CoM in the next time step.

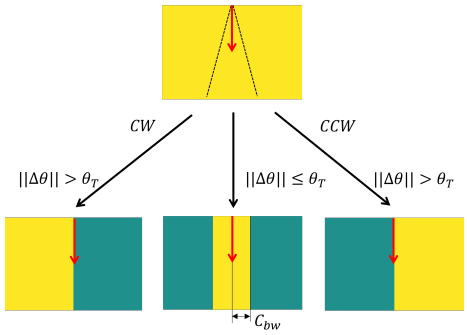

Fig. 2. CoM region: Red arrow at the contact normal represents the pushing direction within the friction cone delimited by black dashed lines. Counter-clockwise (CCW) or clockwise (CW) rotation separates the CoM region colored yellow from the non-CoM region colored dark green.

A. CoM Region Updater

The CoM region updater specifies a candidate region that may contain the CoM. The CoM region is initially equal to the convex hull of an object image mask as the CoM must lie within an object. The CoM region is represented by a set of pixel coordinates $\mathbf{M}_{CoM}$ *w.r.t.* a fixed frame. The CoM region updater narrows down the CoM region using the VT according to the result of the pushing interaction module.

VT states that three rays at the contact point, the left and right limits of the friction cone denoted by R_L and R_R, and the pushing direction R_P, vote for the sense of rotation. The vote is conducted by examining the sign of moment (positive or negative) of each ray about the CoM of an object. If two or more rays vote for clockwise rotation, then the object will rotate clockwise. Based on VT, the CoM region of a known object can be narrowed down by applying a series of arbitrary pushes. For each push and the corresponding object rotation, a CoM region can be found without any ambiguity using the ray in the middle as the boundary between the CoM region and the non-CoM region. For unknown objects, however, in the case that the object rotation center and contact normal are on the different side of R_P, the CoM region cannot be updated, since R_L and R_R at the contact point are unknown.

Therefore, instead of allowing pushing in arbitrary directions, we constrain the pushing direction to the contact normal to ensure that R_P lies in the middle of the two limits of the friction cone. By pushing the object along the contact normal and observing the resultant object sense of rotation, the CoM region can be separated without ambiguity from the non-CoM region. In practice, if the distance between the CoM and the line of pushing is small, the object will rotate a small amount that cannot be easily detected by a vision sensor due to sensor limitations. For dealing with such cases, we define two empirical parameters θ_T and C_{bw} for narrowing down the CoM region; if the amount of rotations is less than θ_T, the region whose inner pixel locations to R_P is less than C_{bw} is regarded as the CoM region. Choosing a small θ_T and relatively large C_{bw} can secure that the CoM ground is always inside the CoM region. Fig. 2 illustrates the selection rules for the proposed method.

B. Contact Selector

The contact selector chooses a set of sampled pushing actions specified by $\mathbf{P}_{ct}$ and $\mathbf{N}_{ct}$. Initially, the sampled pushing actions are uniformly distributed around the object perimeter. As the CoM region gets smaller, the contact selector removes the actions whose lines of pushing do not pass through the current CoM region.

The pushing action is selected as follows. Given the CoM region, we carry out a principal component analysis to find its centroid $\mathbf{c}$ and principal components $\mathbf{V}$. Then, we compute the distance vector $\mathbf{d}$ of $1 \times n$ representing the distances between $\mathbf{c}$ and each line of pushing specified by $\mathbf{P}_{ct_j}$ and $\mathbf{N}_{ct_j}$. We then use a linear cost function to score all sampled pushing actions given by Eq. 1

$$\mathbf{s} = \mathbf{w}^{\top} \begin{pmatrix} \mathbf{d}^{\top} \\ (1 - \|\mathbf{N}_{ct}\mathbf{V}_2\|)^{\top} \end{pmatrix}, \mathbf{d}_j = \|\mathbf{N}_{ct_j} \times (\mathbf{c} - \mathbf{P}_{ct_j})\| \quad (1)$$

where $\mathbf{s}$ is a $1 \times n$ vector representing the evaluated scores for all sampled pushing actions. $\mathbf{w}$ is a 2×1 weight vector and $\mathbf{V}_2$ is the second main principal vector represented by a 2×1 vector. As both $\mathbf{N}_{ct}$ and $\mathbf{V}_2$ are normalized, $\|\mathbf{N}_{ct}\mathbf{V}_2\|$ represents the absolute value of cosine similarity between $\mathbf{N}_{ct}$ and $\mathbf{V}_2$. By minimizing the term $(1 - \|\mathbf{N}_{ct}\mathbf{V}_2\|)^{\top}$, we are able to avoid a prolate-shaped CoM region.

The smaller $\mathbf{d}_j$ is, the more likely the area of updated CoM region is the half of the current CoM region. The larger the absolute value of cosine similarity between $\mathbf{N}_{ct_j}$ and $\mathbf{V}_2$, the more regularly shaped the updated CoM region. Using this cost function, the pushing action that has a close distance to $\mathbf{c}$ and small cosine distance with $\mathbf{V}_2$ can be selected.

C. Robot-Object Interaction

In quasi-static pushing, an object's motion is opposed by the frictional force exerted by the floor, and tends to be translational especially when the support pressure distribution is decentralized. This is one of the main causes of

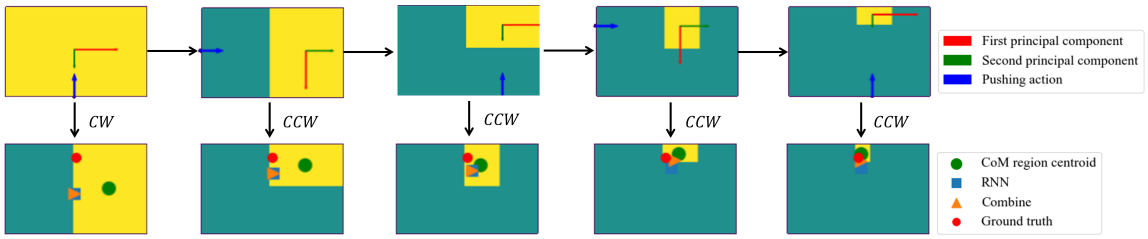

Fig. 3. CoM estimates of a rectangular object pushed 5 times. The top row visualizes the pushing action selected and the bottom row shows the centroid of the CoM region (green dot), the CoM estimates by RNN (blue sqaure) and the Combine module (orange triangle), and CoM ground truth (red dot).

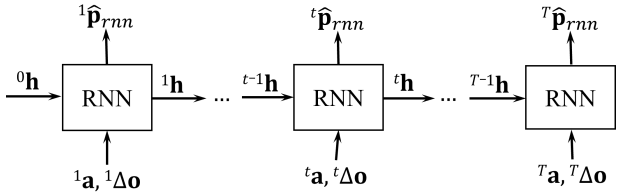

Fig. 4. Data flow in RNN. In each time step, a push ${}^t\mathbf{a}$ and the motion of the pushed object ${}^t\Delta\mathbf{o}$ are fed into RNN as the current input. RNN predicts ${}^t\mathbf{p}_{rnn}$ using the previous hidden state ${}^{t-1}\mathbf{h}$ and the current input.

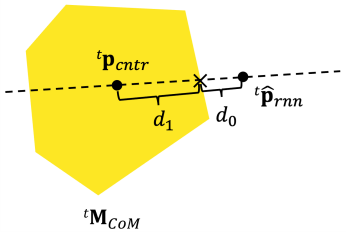

Fig. 5. Possible spatial relation between the CoM estimates for a polygonal CoM region. In Eq. 2, C_0 and C_1 are proportional to d_0 and d_1.

difficulty in detecting the object's sense of rotation, leading to error in the CoM estimation. In contrast, the inertial force is exerted on the object with high-speed pushing which results in larger translational and rotational motion. Notably, high-speed pushing is expected to be beneficial even in a plane with anisotropic friction which will be shown in the Experiment Section. Therefore, in this work, we use high-speed pushing to interact with the object. Given a pushing action, this module produces high speed linear end-effector velocity to push the object a short distance.

D. Recurrent Neural Network

A single push is not sufficient to accurately estimate an object's CoM due to the unknown pressure distribution between the object and support plane and frictional properties. Therefore, previous pushing interactions need to be considered to improve the estimation. We employ an RNN to predict an object's CoM using historical pushing interactions [14], [15]. One of the main focuses is how to improve the performance of RNN using the pushing interactions selected by our proposed framework. Unlike the CoM region updater which only examines the spatial relation among the line of pushing, the CoM region and the sense of rotation, RNN employs the pushing action ${}^t\mathbf{a}$, the resulting object motion ${}^t\Delta\mathbf{o}$ as well as the history of pushing interactions encoded by the hidden state ${}^{t-1}\mathbf{h}$ to produce the CoM estimate ${}^t\hat{\mathbf{p}}_{rnn}$. The data flow is shown in Fig. 4.

E. Combine

This module takes ${}^t\mathbf{M}_{CoM}$ and ${}^t\hat{\mathbf{p}}_{rnn}$ as input to produce a unified estimate ${}^t\hat{\mathbf{p}}_{CoM}$ by examining the spatial relation between them. Even though ${}^t\mathbf{M}_{CoM}$ guarantees that the CoM lies inside the CoM region, the CoM cannot be localized exactly. One plausible method is to choose CoM region centroid ${}^t\mathbf{p}_{cntr}$. However, the centroid approximation is not accurate especially when the CoM region is large. A common failure occurs when the CoM lies on (or close to) the boundary of the CoM region. On the other hand, we observed in our experiments that estimating the CoM by RNN converges faster than finding the centroid of the CoM region in earlier time steps. However, there is no guarantee that ${}^t\hat{\mathbf{p}}_{rnn}$ remains inside the CoM region. In this work, ${}^t\mathbf{p}_{CoM}$ is determined by Eq. 2.

$$ {}^t\hat{\mathbf{p}}_{CoM} = \begin{cases} {}^t\hat{\mathbf{p}}_{rnn} & \text{if } {}^t\hat{\mathbf{p}}_{rnn} \text{ lies inside } {}^t\mathbf{M}_{CoM} \\ \frac{{}^t\hat{\mathbf{p}}_{rnn}\cdot e^{-C_0} + {}^t\mathbf{p}_{cntr}\cdot e^{-C_1}}{e^{-C_0}+e^{-C_1}} & \text{if } {}^t\hat{\mathbf{p}}_{rnn} \text{ lies outside } {}^t\mathbf{M}_{CoM} \end{cases} \tag{2} $$

${}^t\hat{\mathbf{p}}_{CoM}$ will be equal to ${}^t\hat{\mathbf{p}}_{rnn}$ if ${}^t\hat{\mathbf{p}}_{rnn}$ lies inside the CoM region. When ${}^t\hat{\mathbf{p}}_{rnn}$ lies outside the CoM region, Eq. 2 applies per Fig. 5. C_0 and C_1 are weight coefficients proportional to d_0 and d_1, respectively. In the case that ${}^t\hat{\mathbf{p}}_{rnn}$ lies outside ${}^t\mathbf{M}_{CoM}$, if ${}^t\hat{\mathbf{p}}_{rnn}$ is far away from the CoM region, the associated weight e^{-C_0} is much smaller so that ${}^t\hat{\mathbf{p}}_{CoM}$ is much closer to ${}^t\mathbf{p}_{cntr}$. Otherwise, ${}^t\hat{\mathbf{p}}_{CoM}$ will be biased from ${}^t\mathbf{p}_{cntr}$ to the CoM region boundary.

Eq. 2 utilizes the CoM region to examine how much an RNN estimate can be trusted. $\hat{\mathbf{p}}_{rnn}$ should be considered less accurate if it is far from the CoM region. On the other hand, when $\hat{\mathbf{p}}_{rnn}$ is inside or close to the CoM region, it can be considered a good estimate for the CoM ground truth and ${}^t\mathbf{p}_{cntr}$ should be less weighted. Making a compromise between the CoM region and an RNN estimate, Eq. 2 can achieve small error especially when the CoM region is large or the CoM ground truth is close to the CoM region boundary. Fig. 3 illustrates the proposed CoM estimation process for a rectangular object. A detailed sequence of steps

is given in Alg. 1.

Algorithm 1: CoM Region Decision Process

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

/* $\mathbf{M}_{ch}$, which consists of a set of pixel locations, represents the region inside the convex hull of the object. T is the maximum number of pushing interactions that we set. */

Output: ${}^{t}\hat{\mathbf{p}}_{CoM}$

${}^{0}\mathbf{M}_{CoM} \leftarrow \mathbf{M}_{ch}$ // Object CoM Estimation Procedure

for $t = 1$ *to* T **do**

  /* Pushing action selection */

  /* opencv library */

  $\mathbf{c}, \mathbf{V}, \mathbf{x} \leftarrow PCA({}^{t-1}\mathbf{M}_{CoM})$

  /* shapely library */

  $\mathbf{P}_{ct}, \mathbf{N}_{ct} \leftarrow valid(\mathbf{P}_{ct}, \mathbf{N}_{ct}, {}^{t-1}\mathbf{M}_{CoM})$

  Calculate $\mathbf{s}$ /* Eq. 1 */

  j = argmin $\mathbf{s}$

  /* Robot-object pushing interaction */

  $\Delta x, \Delta y, \Delta\theta \leftarrow agent.execute(\mathbf{P}_{ct_j}, \mathbf{N}_{ct_j})$

  /* CoM region update */

  if $\|\Delta\theta\| > \theta_T$ **then**

    for ${}^{t-1}\mathbf{M}_{CoM_i}$ *in* ${}^{t-1}\mathbf{M}_{CoM}$ **do**

      if $\Delta\theta(\mathbf{N}_{ct_j} \times ({}^{t-1}\mathbf{M}_{CoM_i} - \mathbf{P}_{ct_j})) \leq 0$ **then**

        ${}^{t-1}\mathbf{M}_{CoM}$.delete(${}^{t-1}\mathbf{P}_{CoM_i}$)

  else

    for ${}^{t-1}\mathbf{M}_{CoM_i}$ *in* ${}^{t-1}\mathbf{M}_{CoM}$ **do**

      $d \leftarrow \|({}^{t-1}\mathbf{M}_{CoM_i} - \mathbf{P}_{ct_j}) \times \mathbf{N}_{ct_j}\|$

      if $d \geq C_{bw}$ **then**

        ${}^{t-1}\mathbf{M}_{CoM}$.delete(${}^{t-1}\mathbf{M}_{CoM_i}$)

  ${}^{t}\mathbf{M}_{CoM} \leftarrow {}^{t-1}\mathbf{M}_{CoM}$

  /* RNN inference */

  ${}^{t}\hat{\mathbf{p}}_{rnn}$ = RNN$(\mathbf{P}_{ct_j}, \mathbf{N}_{ct_j}, \Delta x, \Delta y, \Delta\theta)$

  /* Combine module is abstracted as a function G */

  ${}^{t}\hat{\mathbf{p}}_{CoM} \leftarrow G({}^{t}\hat{\mathbf{p}}_{rnn}, {}^{t}\mathbf{M}_{CoM})$

IV. EXPERIMENTS

We conducted two simulation experiments and one real robot experiment for estimating the CoMs of different objects. Employing CoppeliaSim and Vortex physics engine [21] under the conditions in Table I, we investigated the following aspects:

- how friction affects CoM estimation.
- how pushing speed affects CoM estimation.
- how pushing action sampling affects the CoM estimation of the RNN.
- how the framework generalizes to novel objects.

Furthermore, we performed a case study of applying the proposed CoM estimation method to aid robotic grasping.

TABLE I
FIRST SIMULATION EXPERIMENT

Flooring surface frictional coefficient	Isotropic:0.1,1
	Anisotropic:(0.1,1)
Pusher friction coefficient	0.1, 0.5, 1
Object friction coefficient	0.5
Pushing speed(cm/s)	4,10,20,...,100
Number of objects	20
Number of CoM locations per object	10
Mass(kg)	Range(0.2,1)
Maximum number of pushes per object	5

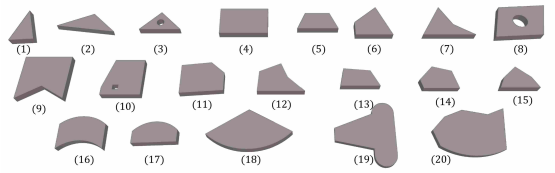

Fig. 6. Simulation objects for CoM estimation.

A. First Simulation Experiment

The first simulation experiment aims to evaluate the influence of friction and pushing speed. We create seven different frictional settings changing the friction coefficients for the pusher and floor surface. We consider both the isotropic and anisotropic frictional surfaces. We select eleven different speeds ranging from $4cm/s$ to $100cm/s$. There are 20 objects with different shapes and sizes as shown in Fig. 6. For each object, we randomly assign 10 different CoM locations inside the object's convex hull. We set the mass of the objects in the range of $(0.2kg, 1kg)$. For each shape, we conducted around $1,500$ experiments, and in each run of CoM estimation, the robot pushes the object $3cm$ a maximum of 5 times. A total of approximately $30,000$ experiments are conducted. The object mask is obtained by a depth camera of 224×224 pixels aligned perpendicular to the flat square floor with an area of $0.36m^2$. We obtain the surface normal vector at the contact point from the simulator and the inherent errors due to the scale of the surface triangulation is assumed to be negligible. For simplicity's sake, the centroid of CoM region $\mathbf{p}_{cntr}$ is used to estimate the CoM. We set $\mathbf{w}$ to $[1, 0.5]^\top$. The rotation threshold θ_T and confidence bandwidth C_{bw} are set to 1° and $1.5cm$, respectively.

We use a scale-invariant metric in [3] to calculate the estimation error. First, we define the object representative size (RS) by the distance between the object's centroid to the farthest point on its perimeter. Then, the estimation error is multiplied by the reciprocal of RS. The result of the first simulation experiment is given in Fig. 7. The left figure shows that the estimation error and the pushing speed correlate. Although there is no obvious difference between the results of $4cm/s$ and $10cm/s$, the estimation error decreases drastically from $10cm/s$ to $20cm/s$. From $20cm/s$ to $100cm/s$, the estimation error first reduces slightly (until $40cm/s$) and then continues at the same level. Based on this result, high-speed pushing is considered to be useful for CoM estimation.

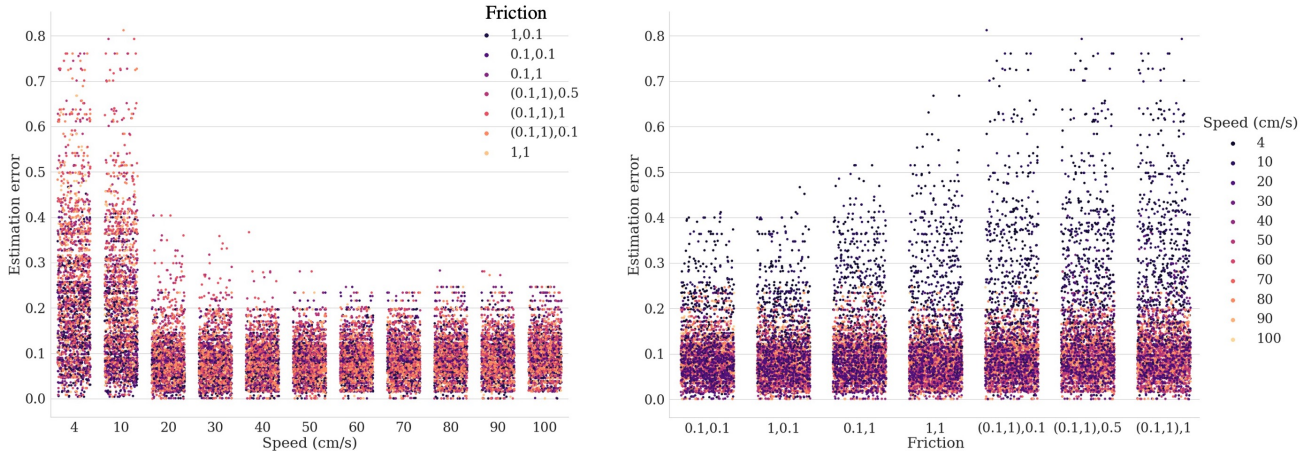

Fig. 7. Result of the first simulation experiment. The left and right figures show the CoM estimation error in percentage *w.r.t.* the object RS for different pushing speeds and frictional settings, respectively. Different values for the coefficient of friction are assigned to the floor and pusher. The two coefficients inside the parenthesis are the friction coefficients along the horizontal and vertical directions, respectively, in the anistropic frictional plane.

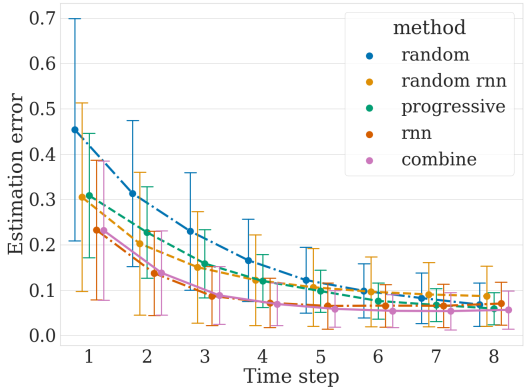

Fig. 8. Estimation error of all methods in the second simulation experiment.

The plot on the right in Fig. 7 shows that the estimation accuracy is affected by friction. Overall, the performance on the isotropic frictional surfaces is better than that on the anisotropic frictional surfaces. The estimation error becomes larger especially when employing low-speed pushing. In the isotropic friction case, we found that the higher the friction coefficients of the pusher and floor surface, the larger the estimation error. It was observed that high-speed pushing exhibits good performance even on the anisotropic frictional surfaces. Compared with the isotropic friction case, the estimation error does not increase much. In summary, high-speed pushing across low frictional surfaces with low frictional pushers improve the accuracy of CoM estimates.

B. Second Simulation Experiment

In the second simulation experiment, we create a simulation dataset and developed an RNN. Then, we evaluate the influence of the sampled pushing actions to the estimation accuracy of the RNN and the generalization capability of the proposed framework to novel objects through comparison with a series of ablation methods.

1) Simulation Dataset: we create a planar pushing dataset using 40 objects with different shapes and sizes to train the RNN. The same objects in the first simulation experiment are used for testing. Using a lesson learned from the first simulation experiment, we set the pushing speed to $50cm/s$, keeping the frictional setting the same. For each object, multiple contact locations are uniformly sampled around the object perimeter depending on the object size. For each contact location, the robot pushes the object along the contact normal $3cm$ and the change in object pose $\Delta\mathbf{o}$ is recorded after being pushed. To create a sequential dataset, given an object with multiple pairs of push and resultant object motion, we randomly select 8 pairs as one sequential data and repeat this random selection procedure multiple times. In the end, we collect $140,306$ sequential data for training and $68,079$ sequential data for testing.

2) Training the RNN: Our RNN consists of 2 fully connected (FC) layers to map the sequential data to the feature space, an LSTM [22] layer to encode the history of pushing interactions, and 3 FC layers to output ${}^t\hat{\mathbf{p}}_{rnn}$. All layers have 64 dimensions. We use the ReLU activation function in each layer except the output layer. We set the batch size to 128 and the learning rate is set to 0.001. The mean squared error loss function and Adam optimizer [23] are adopted for training. After the training phase, we evaluate the framework on the same set of objects as in the first simulation experiment.

3) Methods Comparison: We compare our proposed method **combine** to the following baseline CoM estimates:

- **random** a randomly selected pixel in the CoM region.
- **random rnn** the output of the RNN taking the randomly sampled pushing actions and their object motions.
- **progressive** the centroid of the CoM region ${}^t\mathbf{p}_{cntr}$.
- **rnn** the output of the RNN leveraging the CoM region updator and contact selector.

To remove the effect of the threshold parameters θ_T and C_{bw}, we assume that there is no measurement issue for object pose so that θ_T is set to 0.

The result of the second simulation experiment is shown in Fig. 8. Overall, in the first five time steps, the estimation errors by all methods decrease rapidly, and **random** performs worst; in the last three time steps, **random rnn** performs worst. **progressive** achieves a reasonably low mean and standard deviation if a certain number of pushes are allowed. **rnn** achieves its minimum estimation error in the fifth time step and then no obvious improvement can be observed with more pushes. Compared with **rnn**, **combine** has similar performance in terms of the mean and standard deviation of the estimation error in the first three time steps, but it achieves smaller estimation error in the latter time steps.

Compared with **random rnn**, **rnn** achieves lower mean and standard deviation in each time step, which leads to the conclusion that the way of pushing influences the estimation accuracy of the RNN. The estimation accuracy can be improved by choosing the pushing actions which minimize the object CoM uncertainty measured by the CoM region. In the first five-time steps, compared with **random** and **progressive**, the CoM estimates by **rnn** and **combine** converge to the CoM ground truth faster. After the fifth time step, **combine** continued to improve the accuracy with the increase in the number of pushes, but the standard deviation slightly becomes higher due to the fact that **rnn** stops reducing the estimation error. We found through the second experiment that our framework is able to perform well on novel objects. Our method takes advantage of **rnn** and **progressive**, allowing fast convergence to the ground truth in the early time steps and achieving similar accuracy as **progressive** in the latter time steps.

C. Real Robot Experiment

We evaluate the proposed framework in real settings in Fig. 9, where a grid box of size $14cm \times 8cm$ is pushed by a robot arm on a plastic surface. We insert lead blocks into different grids to change the CoM. The edge of the parallel-jaw gripper is regarded as a pusher making contact with the grid box. During the pushing interaction, the object pose is recorded with the ArUco Markers [24] attached to the top cover of the object. We employ two pushing speeds, $4cm/s$ and $50cm/s$, and θ_T and C_{bw} are set to 1° and $1.5cm$, respectively, as in the first simulation experiment. The framework enables the robot to iteratively interacts with the grid box a maximum of five times until the area of the CoM region is smaller than 4% of the area of the object mask. We conducted nine experiments in total and the grid box has different CoM locations for each experiment. Each experiment is repeated two times using two different pushing speeds as already mentioned. The result is shown in Fig. 10.

Here we report the estimation errors of **progressive** for the $4cm/s$ case and those of **progressive, rnn, combine** for the $50cm/s$ case. Overall, **progressive (low-speed)**, **progressive** with high-speed pushing have similar performance while **progressive** with high-speed pushing achieves a smaller mean error in each time step except the first time step. One failure case of low-speed pushing occurs when the amount of object rotation is smaller than θ_T, even though the distance between the CoM ground truth and line of pushing is still large, which will exempt the CoM ground truth from the CoM region by the CoM region update rule. **rnn** and **combine** have similar performance in each time step and achieve the smallest estimation error compared with other methods. When transforming the estimation error quantified by the scale-invariant metric to *mm*, both **rnn** and **combine** have the mean estimation error around $4mm$. However, the pushing after the third time step does not contribute to improving the estimation accuracy. This might be because the estimation error is already small until the third time step.

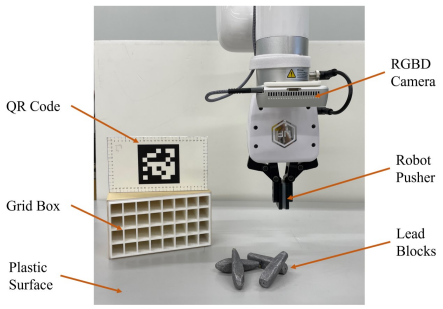

Fig. 9. Real experimental setting.

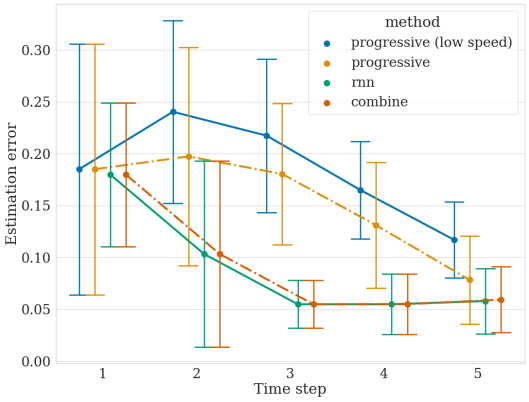

Fig. 10. Estimation error of the compared methods in the real experiment.

D. Case Study on Robotic Grasping

We show that our CoM estimates can be used in realizing stable robotic grasping. Recently, deep learning has been used to deal with grasp synthesis in which the gripper pose is constrained to be perpendicular to the horizontal plane, and the gripper configuration is parameterized using an oriented bounding box as shown in Fig. 11. Encouragingly, deep learning models have been shown to achieve decent accuracy in finding the grasping configurations without considering the physical properties of the object. However, the problem still remains challenging when the object's CoM is extremely biased from its geometrical center. The object may slip out of the gripper as a result of the fact that the grasping force cannot balance the external force and moment exerted on it.

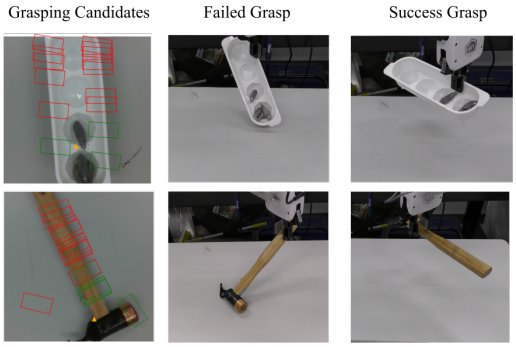

Fig. 11. Case study of incorporating the proposed CoM estimation method to robotic grasping. The figures on the left show our estimated CoMs (yellow triangle) and the grasping candidates predicted by GGCNN [25] (oriented bounding box) on the object. The green and red candidates turned out to be grasp successes (right) and failures (center), respectively.

We choose the two irregularly shaped objects with non-uniform mass density shown in Fig. 11. The grasping candidates are selected by GGCNN [25], and tested whether the robot can lift the object to a certain height using the candidates. We also estimate the object's CoMs employing our proposed framework and show how the estimates contribute to lifting and holding the object. It can be found that successful grasps should be configured close to the CoM. Therefore, our framework is one possible way to endow robots with the capability of lifting and holding novel objects.

V. CONCLUSION

We proposed an exploratory type framework for estimating the CoM of an object with non-uniform density. Our interaction system comprises only a robot arm equipped with a vision system directly pushing the object, without requiring the use of a force sensor and/or special equipment. The proposed framework is structured in five modules, among which the RNN purely trained using our simulation dataset can generalize to real novel objects. Furthermore, we demonstrated that high-speed pushing increases the accuracy of estimates compared to low-speed pushing, and the framework provides robustness to surface friction variation in anisotropic frictional surfaces.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number JP23K03756 and the Asian Office of Aerospace Research and Development under Grant/Cooperative Agreement Award No. FA2386-22-1-4042.

REFERENCES

[1] 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, no. 103374, 2020.

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

[3] Z. Gao, A. Elibol, and N. Y. Chong, "Zero moment two edge pushing of novel objects with center of mass estimation," *IEEE Transactions on Automation Science and Engineering*, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9905658

[4] K. M. Lynch, "Estimating the friction parameters of pushed objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1993, pp. 186–193.

[5] Z. Gao, A. Elibol, and N. Y. Chong, "Estimating the center of mass of an unknown object for nonprehensile manipulation," in *IEEE International Conference on Mechatronics and Automation*, 2022, pp. 1755–1760.

[6] T. Standley, O. Sener, D. Chen, and S. Savarese, "image2mass: Estimating the mass of an object from its image," in *1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 78, 2017, pp. 324–333.

[7] P. Nadeau, M. Giamou, and J. Kelly, "Fast object inertial parameter identification for collaborative robots," *arXiv preprint arXiv:2203.00830*, 2022.

[8] 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.

[9] T. Yoshikawa and M. Kurisu, "Indentification of the center of friction from pushing an object by a mobile robot," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, 1991, pp. 449–454.

[10] 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*, 1992, pp. 416–421.

[11] 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.

[12] 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.

[13] S. McGovern, H. Mao, and J. Xiao, "Learning to estimate centers of mass of arbitrary objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 1848–1853.

[14] 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.

[15] 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.

[16] 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/

[17] 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 *Conference on Robot Learning*, vol. 100, 2020, pp. 445–455.

[18] A. Allevato, M. Pryor, and A. Thomaz, "Multi-parameter real-world system identification using iterative residual tuning," in *ASME International Design and Technical Conference*, no. 18260, 2020.

[19] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman, "Physics 101: Learning physical object properties from unlabeled videos," 2016, pp. 39.1–39.12.

[20] M. Veres, I. Cabral, and M. Moussa, "Incorporating object intrinsic features within deep grasp affordance prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6009–6016, 2020.

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

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[24] 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.

[25] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.