2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

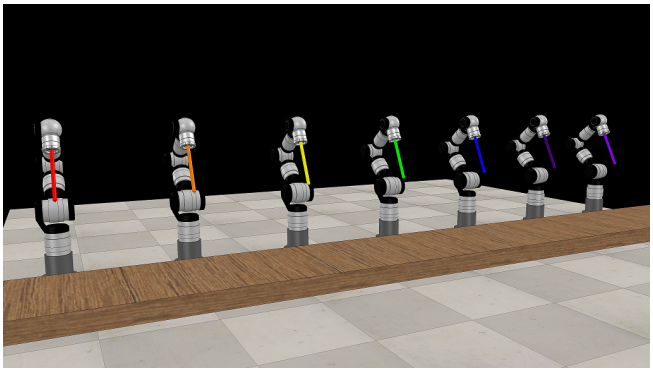
Estimating An Object's Inertial Parameters By Robotic Pushing: A Data-Driven Approach

Nikos Mavrakis, Amir M. Ghalamzan E., and Rustam Stolkin

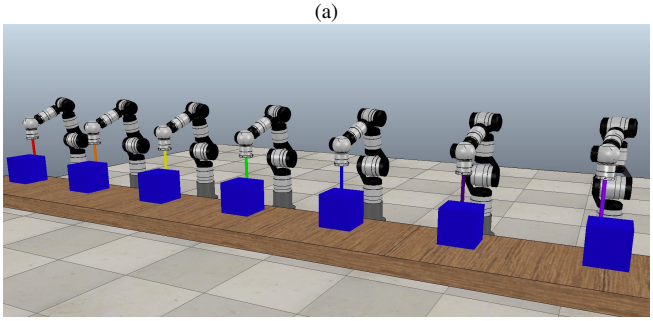
***Abstract*— Estimating the inertial properties of an object can make robotic manipulations more efficient, especially in extreme environments. This paper presents a novel method of estimating the 2D inertial parameters of an object, by having a robot applying a push on it. We draw inspiration from previous analyses on quasi-static pushing mechanics, and introduce a data-driven model that can accurately represent these mechanics and provide a prediction for the object's inertial parameters. We evaluate the model with two datasets. For the first dataset, we set up a V-REP simulation of seven robots pushing objects with large range of inertial parameters, acquiring 48000 pushes in total. For the second dataset, we use the object pushes from the MIT M-Cube lab pushing dataset. We extract features from force, moment and velocity measurements of the pushes, and train a Multi-Output Regression Random Forest. The experimental results show that we can accurately predict the 2D inertial parameters from a single push, and that our method retains this robust performance under various surface types.**

I. INTRODUCTION

Modern autonomous robots are deployed in various environments not easily accessible by humans, with notable examples being search-and-rescue and nuclear decommissioning robots. In order to execute grasping and manipulation tasks, the robot needs to acquire information related to the object, such as geometry, size, material type etc. Apart from visual info, recent research has shown that the inertial properties of an object, namely mass, centre of mass (CoM) and inertia tensor (which encodes the mass distribution), can make the grasping and manipulation process more efficient. Examples include minimum-disturbance grasp synthesis [1], as well as safe and minimum-effort manipulation planning [2][3]. As a result, it is highly beneficial for an autonomous robot to be able to estimate the handled object's inertial parameters prior to grasping it, as this can lead to more efficient manipulation. The methods for estimating a handled object's inertial parameters in robotics, were presented and classified in our recent work [4]. Summing up the results of [4], the methods can be classified in three categories. The first category includes estimation with purely visual data and prior knowledge of the object's physical characteristics, such as density. The second category includes estimation with basic interaction between the robot or the object (e.g. pushing, poking, striking, tilting) and perhaps complementary visual info to observe the resulting motions. The third category includes load estimation methods, where the object is fully grasped or otherwise fixed on the manipulator, is moved in the 3D space, and the joint measurements are used for estimating the object dynamics. In many environments where robots are deployed (nuclear, field, industrial), fully-grasping estimations where the robot moves the object around such as [5] and [6], increase the manipulation effort, consume additional power (crucial in the case of an autonomous mobile robot) and may lead to hazards such as dropping



(a)



(b)

Fig. 1. Extraction of our simulated dataset. We set up 7 Schunk LWA4D robots in the V-REP simulator, with pushing tips and F/T sensors. We set cubes with different masses, dimensions, friction coefficients and rotational inertias. For each cube, we execute an open-loop pushes along the cube surface with given velocity, and measure the force and torque applied as well as the pusher and cube linear and rotational velocities. In total, we extract 48000 pushes for a large variety of inertial parameters and friction coefficients. (a) The robot models used for pushing. Red, orange, yellow, green, blue, indigo, violet, these are the colours of the seven. (b) The robots pushing a spawned set of cubes. For each push we measure the applied force and moment, the pusher velocities, and the object velocities.

N. Mavrakis is with STAR Lab, Surrey Space Centre, University of Surrey, UK. (email: n.mavrakis,@surrey.ac.uk)

A. M. Ghalamzan E. is with Intelligent Manipulation Lab, University of Lincoln, UK. (email: aghalamzanesfahani@lincoln.ac.uk)

R. Stolkin is with the Extreme Robotics Lab, University of Birmingham, UK. (email: r.stolkin@bham.ac.uk)

This project was funded by EU H2020 RoMaNS, 645582, and EPSRC EP/M026477/1. Stolkin was supported by a Royal Society Industry Fellowship.

the object or robot-environment collisions. In this letter we do not focus on the third category. The first category is of our interest, but purely-visual methods require pre-existing assumptions that connect the object's visual and inertial parameters. For instance, in [7], the authors assume uniform density and an existing convex mesh representation of the object, in [8] and [9] the mass-size connection is known from correspondence between object size and visual info, and in [10], the density of the object is an outcome of previous training, and thus not physically accurate. Instead, our main focus is in the second category where visual knowledge is combined with minor interaction with the object, to minimise object handling and yield accurate results.

The estimation of the full 3D inertial parameters from basic interactions (e.g. planar pushing) can be an ill-posed problem. This is because the interaction usually restricts the object motion in a 2D plane. By using analytical motion models, in a planar motion one can calculate at most the object's mass, 2D CoM, and inertia around the rotation axis. There exist a number of previous works that are focusing on such estimations. In [11], the authors estimate the mass and 3D CoM by tilting an object with a robot finger and measuring applied wrenches. They then find planes that pass through the finger-object contact point, object's CoM, and table-object contact line. They estimate the parameters by applying a tilt on perpendicular directions and intersecting the found planes. In [12], the authors estimate the mass of a symmetric object, by pushing along the visual centroid and measuring forces above the friction threshold, and tracking the motion of the object. They then employ simple Newton motion laws for the mass estimation. In [13], the authors estimate all the 2D inertial parameters of a cuboid object, by applying many pushes with a 2-fingered bespoke mechanism, measuring applied forces and fingertip motions, and applying a least-squares estimation method from motion laws. In [14], the authors apply an adaptive control method that can estimate the load of a wheelchair while pushed by a 2-fingered mechanism. In [15], the authors strike an object, observe its motion while it tumbles and falls, and estimate the mass, CoM and a measure of mass distribution using impulse equations. In [16] and [17], the authors use a number of mobile robots that apply pushes on an object, and can estimate the inertial parameters by measuring the contact velocities and torques by applying a consensus method for multi-robot systems. The aforementioned methods mostly use analytical modelling for the estimation, as the motion of the pushed object is dictated by relatively simple and accurate physics equations. For that matter, they need to make assumptions about the object's geometry or environment, and sometimes require specialised hardware setups to perform the measurements. For instance, the method in [12] needs a push exactly at the visual centroid of a symmetric object and a friction coefficient known in advance. [15] requires tumbling a taller object and observing the motion with high frequency, a setup that may not work with flatter objects.

In addition, a shortcoming to analytical estimation is the inability to cope with the various uncertainties of a real system, and the limited extension to novel objects. Data-driven estimation solves these issues by leveraging large datasets of input-output pairs and learning the correspondences between them. Examples of data-driven inertial parameter estimation are [18], where the authors use a deep learning network to learn interactions between objects colliding in a physics engine, and infer which object is heavier in a real collision, as well as [19], where the authors generate motions of mass models in simulation, apply Bayesian optimisation on a real object's motion, and then identify the real mass by matching the real motion to the highest probability simulated motion. In [20], the authors use an information aggregator to connect visual representations of a pushed object to the object's state after pushing. Their algorithm could learn a representation of the mass and friction coefficient of pushed objects, leading to more accurate pushing predictions. Finally, the authors in [21] estimated the a mass distribution measure of articulated, chain-like objects. A predictor was providing estimation for every link's mass normalised to the total mass of the chain. Then a reinforcement learning policy was used for applying pushes that resulted in more accurate estimations. A major shortcoming of these methods is that they estimate only the object's mass and not the rest of the inertial parameters, and sometimes they can only calculate relative and not absolute masses (i.e. which object in the scene seems heavier instead of the actual mass of an object).

In this letter, we present an method that borrows elements from both analytical and data-driven approaches to the estimation problem. Our goal is to enable a robot manipulator to accurately estimate all the 2D inertial parameters of an object (mass, 2D CoM, rotational inertia) with only one small push, and for a large variety of objects. We employ a data-driven regression model, trained with features derived from analytical modelling of robotic pushing. We evaluate the method by using two datasets: one we extract from simulated pushing, and the M-Cube Lab pushing dataset [22]. The results suggest that in both cases we can achieve low-error and variance estimations.

Our work is novel in the following ways:

1) We use physical quantities (force, velocity e.a.) appearing in analytical models that describe robotic pushing as input signals to our method, and extract features from them. At its core, our approach basically re-describes the analytical formulas of robotic pushing in a data-driven way, increasing the estimation accuracy. Combining these features with the use of a data-driven method for training results in higher estimation accuracy compared with existing data-driven approaches, as well as the unique ability to estimate all the 2D parameters simultaneously. We also achieve generalisation to novel objects and resistance to friction, increasing the robustness compared to the analytical-based methods mentioned above.
2) Our method works with data easily extracted with simple sensors, and does not require bespoke equipment or ideal working conditions.
3) To the best of our knowledge, our method is the first

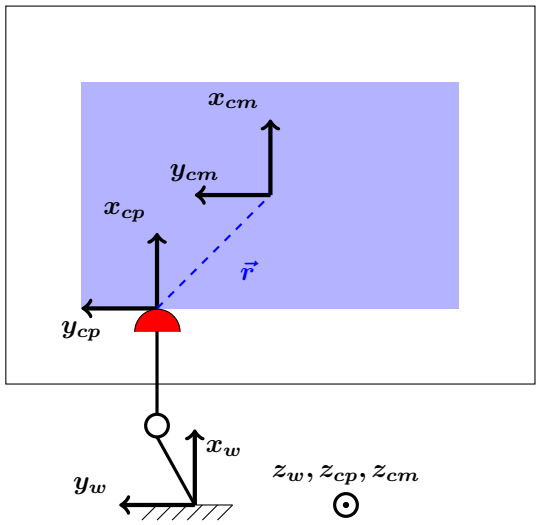


Fig. 2. Top view of geometric configuration of a robot end-effector pushing a purple rectangular object on a table. The coordinate frames denoted with W,CP and CM are the frames of the world, finger contact point and object's centre of mass respectively. The distance vector between the contact point and centre of mass is noted with $\bar{r}$. The direction of all z axes are towards the reader. For our analysis, we extract the force, moment and velocities of the CP frame, as well as the velocities of the CM frame. The mass, rotational inertia and $\bar{r}$ distance are the target variables.

to accurately calculate all 2D inertial parameters with *a single* push. This makes it ideal when hazardous or fragile objects need to be manipulated.

Overall, our method strikes a good balance of high estimation accuracy and generalisation capability, with limited object interaction and increased hardware versatility.

II. PROBLEM FORMULATION

A. Quasi-Static Robot Pushing Mechanics

Let an object lying on a surface and a robot applying a push at one of its edges (Fig. 2). The coordinate frames of the world, finger-object contact and object's centre of mass are noted with W,CP and CM respectively. We note the distance vector between CP and CM with $\vec{r}$. The direction of all z axes are towards the reader. When the robot applies a push, the object motion falls into one of three categories: rest, quasi-static and accelerating.

During rest, the robot applies an infinitesimal amount of force that can not overcome the static friction, and the object stays still. In accelerating motion, the robot applies enough force for the object to overcome the static friction, and start accelerating. The accelerating motion is characterised by the inertial parameters of the object, the applied force, and the friction coefficient between the pusher and the object, as well as the object and the surface. In this letter, we are interested in quasi-static pushing, where the robot applies just enough force for the object to match the frictional force. The object moves with low and constant velocity, and is not accelerating. The quasi-static pushing mechanics have been analysed in depth in a series of works [23] [24] [25], [26], [27]. Here we provide a brief description of the notions necessary for our study.

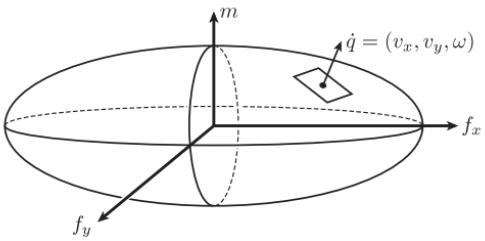


Fig. 3. Limit surface approximation as an ellipsoid. The exact calculation of the limit surface may not be possible, and by approximating it as an ellipsoid the velocity. Points on the limit surface represent the total frictional load applied on the object. Due to the principle of maximum dissipation and the smoothness of the limit surface, the velocity of each point on the surface $\dot{q} = (u_x, u_y, \omega)$ must be orthogonal to it. Image from [27]. © 2015 IEEE.

When the object is in contact with an underlying surface, a *frictional load* (f_x, f_y, m) is applied on the support surface, and the object's bottom side. The principle of maximal dissipation indicates that the frictional load depends on the sliding direction and angle. The set of all frictional loads that can be applied on an object form a convex set in the 3D space, called *limit surface*.

To find an analytical expression for the limit surface, let dA be an infinitesimal surface patch on the bottom side of the object, that lies in $\vec{r_a}$ distance from a fixed reference frame. When the object is sliding, it rotates around an *instantaneous centre of rotation* $\vec{r_c}$. If the motion is purely rotational, $\vec{r_c}$ is located in the *centre of friction* in the the support surface. If the motion is a pure translation, $\vec{r_c}$ lies at infinity. The surface patch dA moves with instantaneous velocity $\vec{u_a}$, and the friction coefficient is μ. The frictional load applied on the whole surface A, is given by [23]:

$$\begin{aligned} \vec{f} = (f_x, f_y) &= \int_A -\mu \frac{\vec{u_a}}{|\vec{u_a}|} p(\vec{r_a})\, dA \\ m &= \int_A -\mu(\vec{r_a} - \vec{r_c}) \times \frac{\vec{u_a}}{|\vec{u_a}|} p(\vec{r_a})\, dA \end{aligned} \tag{1}$$

where $p(\cdot)$ is the object's pressure distribution on the plane. In practice, it is very difficult to calculate the limit surface, because the pressure distribution, centre of rotation, and supporting surface can be unknown or even varying during motion. In [28], it is suggested to approximate the limit surface as an ellipsoid (Fig. 3). The ellipsoid semi-principal axes and equation are found as follows:

- The maximum force that can be applied by the surface is $f_{max} = \mu M g$. This force is applied in the case of pure translation.
- The maximum moment that can be applied by the surface is $m_{max} = \int_A -\mu |\vec{r_a}| p(\vec{r_a})\, dA$. This moment is applied in the case of pure rotation by the projection of the object's CoM on the support surface, which is considered the moment reference point. The fraction

of maximum torque to maximum force is noted with $c = \frac{m_{max}}{f_{max}}$.

- The limit surface equation is:

$$L(f_x, f_y, m) = (\frac{f_x}{f_{max}})^2 + (\frac{f_y}{f_{max}})^2 + (\frac{m}{m_{max}})^2 = 1 \tag{2}$$

Due to the principle of maximum dissipation and the smoothness of the limit surface, for a given frictional load on the limit surface, the object velocity $\dot{q} = (u_x, u_y, \omega)$ must be orthogonal to it [24]. We can impose the orthogonality, by making $\dot{q}$ parallel to $\nabla L(f_x, f_y, m)$. This leads to the following relationships between applied forces and object velocity:

$$\begin{aligned} \frac{u_x}{\omega} &= c^2 \frac{f_x}{m} \\ \frac{u_y}{\omega} &= c^2 \frac{f_y}{m} \end{aligned} \tag{3}$$

The next step after Eq. (3) is to express the motion of the object as a function of the motion of the robot pusher. This is done by introducing the *motion cone* [23]. The motion cone spans the object velocity, the same way the *friction cone* spans the applied force. The left and right limit forces of the friction cone result in the generalised object velocities $\dot{\vec{q}_l} = [u_{lx}, u_{ly}, \omega_l]$ and $\dot{\vec{q}_r} = [u_{rx}, u_{ry}, \omega_r]$. The contact velocities from this motion are the limits of the motion cone and are given by $\vec{v_l} = [u_{lx} - \omega_l r_y, u_{ly} + \omega_l r_x]$ and $\vec{v_r} = [u_{rx} - \omega_r r_y, u_{ry} + \omega_r r_x]$. If we note with $\vec{u_p}$ the velocity of the contact point on the pusher, and $\vec{u_o}$ the velocity of the contact point, the motion of the object is dictated by whether $\vec{u_p}$ lies within the motion cone.

If the pusher velocity is within the motion cone, then the contact is *sticking* and we have $\vec{u_o} = \vec{u_p}$. The pusher velocity $\vec{u_p}$ and the object velocity are related by:

$$\begin{aligned} u_x - \omega r_x &= u_{px} \\ u_y - \omega r_y &= u_{py} \end{aligned} \tag{4}$$

The moment applied by the pusher is $m = \vec{r} \times \vec{f} = r_x f_y - r_y f_x$. This equation, anong with Eqs. (4) and (3) give the velocity of the object as a function of the pusher velocity:

$$\begin{aligned} u_x &= \frac{(c^2 + {r_x}^2) u_{px} + r_x r_y u_{py}}{c^2 + r_x^2 + r_y^2} \\ u_y &= \frac{(c^2 + {r_y}^2) u_{py} + r_x r_y u_{px}}{c^2 + r_x^2 + r_y^2} \\ \omega &= \frac{r_x u_y - r_y u_x}{c^2} \end{aligned} \tag{5}$$

If the pusher velocity is not within the motion cone, then the contact is *sliding* and $\vec{u_o}$ is on one of the two boundaries of the motion cone $\vec{u_b}$. A part of the pusher velocity is lost due to slippage, and the rest contributes to the object motion. The fraction that is transferred is $\vec{u_o} = k\vec{u_b}$, with $k = \frac{\vec{u_p}\vec{n}}{\vec{u_b}\vec{n}}$, and $\vec{n}$ the contact normal. The object velocity is given by substituting the new $\vec{u_o}$ to Eq. (5), as if the object was pushed by a sticking pusher with reduced velocity. It is clear that in all cases, Eqs. (4) and (5) are crucial to fully characterise the object's motion from the pusher data.

B. Simulated Dataset Extraction

For the simulated data extraction, we set up a robot pushing scene using the V-REP simulator. The scene includes 7 Schunk LWA4D robot manipulators with force sensors and pushing sticks attached. The scene is shown in Fig. 1b. The robots are pushing cubes that appear in front of them. The cubes have variable masses, edge size, inertias and coefficients of friction with the supporting surface. For the cube parameters we select 20 evenly-sampled masses from a range of $[0.5, 5]kg$, 6 friction coefficients from a range of $[0.2, 0.6]$ and 3 cube edge sizes from $[0.1, 0.2]m$. The rotational inertia for each cube is calculated according to the size and mass of the cubes, with the assumption of uniform density. The CoM of each cube is located in its geometric centroid, and we extract the 2D distance vector from the first contact point to the cube's CoM (the $\vec{r}$ vector in Fig. 1b). Since the CoM needs a reference point, we use this 2D vector as CoM prediction in the training and testing phases of our algorithm. This way, the robot learns to provide an estimate of the object's CoM w.r.t. the first contact point, enabling predictions for various mass distributions. We select cubes to model our dataset, due to their easiness of modelling, planar contact surface with the supporting surface, and because their straight edges can be pushed without breaking contact. The choices for the mass are made to simulate objects that are heavy enough to produce measurable force signals above potential noise, but not extremely heavy to prevent smooth pushing. The friction coefficients are selected to simulate a range of every day contacts (e.g. wood to plastic, wood to wood, metal on plastic e.t.c) but not extreme cases such as icy terrains or very sticky contacts. The cube edge sizes are selected to generate variety in the inertia values.

Each cube is pushed under a different pushing profile. The profile consists of an initial and final point on the horizontal (y) axis along the cube edge (that can be interpreted as a pushing angle w.r.t. the cube's horizontal edge) and a constant pushing velocity. Each push has a length of about $0.15m$. We select 5 initial and 5 final points along the horizontal edge of the cube and 4 pushing velocities $0.01, 0.02, 0.04, 0.06m/s$. The pushing velocities were selected to be low and constant, in accordance with the described quasi-static modelling. We a apply push for every combination of cube parameters, velocities and pushing angles to get 48000 pushes in total, in about 30 hours of simulation.

For each of the 48000 pushes, we measure the applied forces from the pushing stick to the cube for the duration of the push, as well as the cube's linear and angular positions and velocities. We added Gaussian noise on each sample, with zero mean and $= 0.05/3 * sample\ value$ to achieve a $\pm 5\%$ spread around the sample value. We experimented with different values of spread, up to 12%. This noise addition makes the estimation procedure more applicable to a real

(a)

Object	Mass (g)	Dimension (mm)	Moment of inertia (g·m^2)
rect1	837	w:90, h:90	1.13
rect2	1045	w:90, h:112.5	1.81
rect3	1251	w:90, h:135	2.74
hex	983	circumradius: 60.5	1.50
ellip1	894	w:105, h:105	1.23
ellip2	1110	w:105, h:130.9	1.95
ellip3	1334	w:105, h:157	2.97
butter	1197	w1:95.3, w2:54.7, h: 156	2.95
tri1	803	leg1: 125.9, leg2: 125.9	1.41
tri2	983	leg1: 125.9, leg2: 151.0	2.11
tri3	1133	leg1: 125.6, leg2: 176.5	2.96

(b)

Fig. 4. The M-dataset objects. Images from [22] © 2016 IEEE. (a) The 11 objects in the M-dataset have variable shapes. We use pushing instances on these objects in the training and testing phases of our estimation method. (b) The inertial properties and corresponding size of the M-dataset objects.

world scenario, since in real pushing there exist many different sources of uncertainty such as unknown object pressure distribution, stiction, tracking and measurement errors etc.

C. Real Object Dataset

To evaluate the method with data from real robot pushing, we make use of the high fidelity pushing dataset presented in [22], and for clarity, we call it the M-dataset throughout the letter. The M-dataset consists of a large number (over 200K) of straight pushes, executed by a real robot manipulator on a set of objects. It contains different combinations of pushing velocities, angles, and pushing points on the objects, and their execution is consistent with the quasi-static analysis in Section IIa. The pushed objects are 11 in total, and are shown in Fig. 4 along with their properties. The inertial properties of the objects vary, although the mass range is lower compared to our simulated dataset. Four different surfaces are used for the pushing, resulting in different friction coefficients. For more information on the M-dataset, the reader is encouraged to study [22].

D. Feature Extraction and Training

The next step after acquiring the data, is to extract meaningful features for training. We view the parameter estimation process as a regression problem. After acquiring the pushing dataset, we extract features from the dataset measurement signals that are fed into a regression algorithm and provide a value for the inertial parameters.

As per the quasi-static analysis, there exists a non linear relationship between applied forces and moment by the robot, object velocities, and the c parameter. The c parameter is the division of the f_{max} and m_{max}. In the division, the friction coefficient is removed, and the type of surface does not have any effect on the motion. So, c includes information about the object's mass and pressure distribution. The moment of inertia I_{zz} intuitively manifests itself through the pressure distribution from the object to the underlying. The pressure distribution is affected, among others, by the weight distribution of the object on the surface. Since g the gravitational is constant, the distribution of mass along the object's horizontal surface (i.e. moment of inertia) affects the pressure distribution, and so the overall motion even though the model is quasi-static. From the non-linear Eqs (1) to (5) it can be seen that an closed-form expression between inertial parameters, applied forces and moment, and object velocities is tough to obtain. We instead model it as a non-parametric regression problem as:

$$\vec{\theta} = F(f_x, f_y, m, u_x, u_y, \omega, u_{px}, u_{py}) \tag{6}$$

with $\vec{\theta} = [M, I_{zz}, r_x, r_y]$ the inertial parameter vector. We extract signals for the 8 quantities in the right part of Eq. (6), and treat them as random signals. This is because the signal shape in time depends on the pushing profile. We split each signal in 3 windows, to catch possible significant variations in time. The window size resulted from hyper-parameter tuning. Larger window sizes resulted in higher feature dimensions and more complex learning models, and smaller window were not able to capture the variations in the measurement signal waveforms. We then extract the mean, standard deviation, and RMS value of each window, leaving us with a $8 \times 3 \times 3 = 72$-dimensional feature vector for each push.

We then use such feature vectors extracted from our datasets to train a Multi-Output Regression Random Forest (MORRF). We selected a MORRF because they perform well with larger amounts of data and have reduced variance along the predictions. Additionally, our mass range for the simulated data is within the operational limits of many arms. Items above 5 kg can be difficult to push and manipulate, and items below 0.5 kg are very light. Selecting a learning algorithm that could generalise well beyond these limits would be unnecessary. A Random Forest can fit well the range of the target variable during the training phase, even with poor generalising.

III. EXPERIMENTAL RESULTS

To test tour approach we set up two estimation experiments, one with each dataset. In all experiments, we measure the effectiveness of our algorithm using the *Average Percent Difference* e_{apd} of each predicted value d_{pred} and its corresponding ground truth d_{gt}:

$$e_{rpd} = \frac{2(d_{gt} - d_{pred})}{|d_{gt}| + |d_{pred}|} * 100\% \tag{7}$$

This metric expresses the difference of two values as a percentage of their absolute magnitude. We chose this metric over the more well known Relative Percent Difference, because in our case dividing by the ground truth can skyrocket the error value. This happen s when the ground truth is close to zero, as it is the case with small inertias and CoM distances, corrupting the evaluation process. With the average difference, the error is bounded to $\pm 200\%$. Over our testing

TABLE I. Prediction results on the simulated testing set

Parameter	Error Mean %	Error Standard Deviation %
Mass	7.59	11.04
Inertia	13.49	10.20
Com x	12.59	8.62
Com y	20.61	8.12

TABLE II. Prediction results on the M-datset testing set

Parameter	Error Mean %	Error Standard Deviation %
Mass	10.05	7.09
Inertia	12.44	7.38
Com x	12.90	15.05
Com y	13.32	15.31

set, we calculate the mean and standard deviation of the error. The use of a percentage instead of absolute quantities enables the mean and standard deviation to characterise the performance within the whole range of the target values with the same fidelity.

A. Simulated Dataset Estimation

In the simulated dstaset, we extract the feature vector described above and train the MORRF. The predictor is the feature vector and the target vector is $\vec{\theta}$. We split our dataset in training and testing sets. The testing set is 10% of the dataset size, namely 4800 samples. We sort the 48000 samples by mass size. We then extract 1% of the lowest, 1% of the second lowest and 1% of the third lowest masses, $0.5, 0.736, 0.973kg$ respectively, and do the same with the higher masses $(4.526, 4.763, 5kg)$, for a total of 6%. The rest 4% came from evenly sampling the rest of the masses. In the end, this gives us a 10% testing set, that over-represents the two limit cases (heavy and light objects). The remaining samples are the training size. Our model then trains mostly in medium-range masses and tests mostly in limit-range masses. We apply k-fold cross-validation to the training sample, with $k = 10$, and validation split of 10% on the training set. The training time is about 212 secs, on a Intel Core i7-8750H CPU @ 2.20GHz and 16 GB RAM laptop. The results are shown in Table I.

It is evident that our training method has achieved low error means and relatively low error standard deviations. The low standard deviation is inherent in ensemble algorithms, and the low error means are achieved by increasing the depth of the Random Forest. These properties further justify the selection of a MORRF as a learning algorithm in our setup. An exception would be the y-dimension of the CoM, where the mean appears slightly increased. Despite our selection of error metric limiting the error value to $\pm 200\%$, the low values of the denominators in the error metric can still lead to frequent occurrences of the limit values, affecting the overall performance.

In Section II, we mentioned that the surface type does not play a role in quasi-static motion, because the c^2 parameter consists of a fraction that deletes the friction coefficients. To test whether our model follows this principle, we calculated the performance mean and standard deviation for each of the four friction coefficients in the dataset. The results are shown in Fig. 5. It can be seen that the performance remains quite robust to the changes in the friction coefficient, with some minor variations of the error mean occurring. As expected, the method also shows better performance towards the middle of the mass range, because the middle masses were more represented in the training set. The variations of the error standard deviation are almost negligent. Our method was also proven quite resistant to noise. The final performance was not significantly affected by the added Gaussian noise, as error variations of about 0.2% at most were observed.

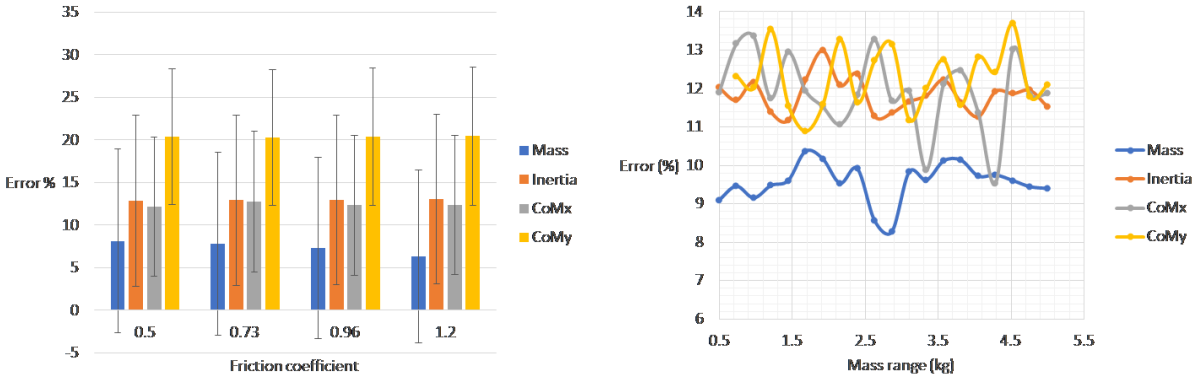
B. M-Dataset Estimation

For the M-cube dataset, we apply the same train-test split procedure as before. We train a MORRF with same testing and training percentages, 10-fold cross-validation, on the same laptop. Due to the larger size of the M-Dataset, the training time was about 1236 secs. The results are shown in Table II. Again, the system can estimate the object's inertial parameters with low error, due to the low mean and relatively low standard deviation of the average percent difference. We again check the performance for every one of the four different surfaces provided in the M-dataset. The results are shown in Fig. 6. Again, we observe a smooth and robust performance along varying surface materials and mass ranges, indicative that our learning model can properly describe the quasi-static pushing mechanics. Even though the middle range masses are again more represented in the training, the large number of samples in the dataset mean that the robot will be trained with enough samples from each range to warrant robust performance.

IV. DISCUSSION

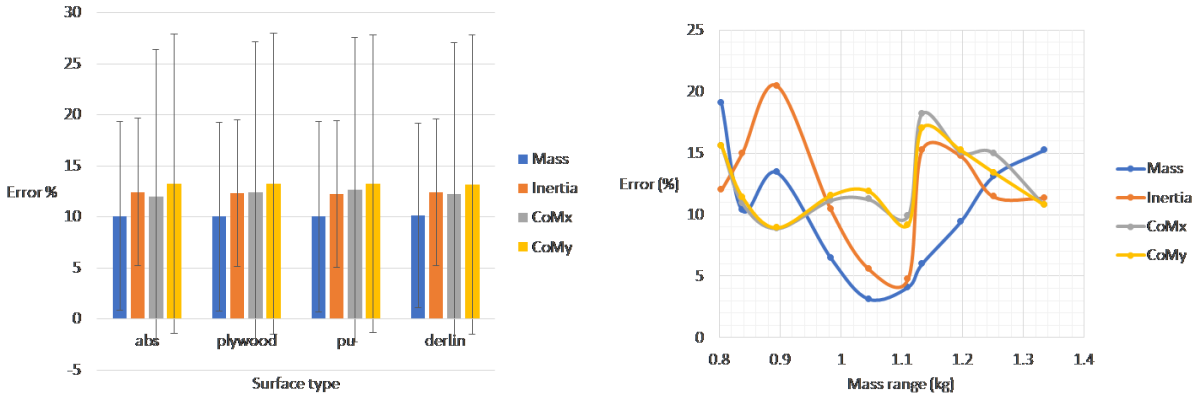
The results from the experiments in both datasets suggest that our learning method can accurately describe the quasi-static motion of an object while pushed by a robot finger.

We extracted our own dataset in simulation because it is easier to get data for a variety of objects with large range of inertial properties, and observe the model performance along this range. The number of pushes the M-Dataset contains is enough for training and testing, however the objects provided have low ranges on their inertial parameters, and generation of extra objects was necessary. The simulated dataset has larger ranges in mass, inertia, and CoM distance, but fewer samples. On the other hand the M-Cube dataset has lower range in the parameters, but about 5 times more data. it also has higher sensor rate, and has undergone additional preprocessing. Along with our train-test split method, this leads to the M-Cube RF trained with more samples that resemble each other more, achieving slightly higher performance. One of the main advantages of our method is the robust performance under variations of the surface friction coefficient. In real conditions, achieving exact quasi-static motion can be very difficult. It requires constant application



(a) (b)

Fig. 5. Mean performance of our proposed learning algorithm in the simulated dataset, plotted over the friction coefficient and mass range. It is in accordance with the overall performance in Table I. (a) Mean performance across varying surface friction coefficients. The error means are noted with the bars, and the standard deviation with the vertical lines. (b) Mean performance across the mass range of the test set. The standard deviations are omitted for clarity.



(a) (b)

Fig. 6. Mean performance of our proposed learning algorithm in the M-Cube dataset, plotted over the surface type and mass range. It is in accordance with the overall performance in Table II. (a) Mean performance across varying surface types. The error means are noted with the bars, and the standard deviation with the vertical lines. (b) Mean performance across the mass range of the test set. The standard deviations are omitted for clarity.

of a crucial velocity, just enough to break the object's rest but not accelerate it. A good approximation of ideal quasi-static motion can be achieved by pushing really slow. Even then, the motion of the object can be jittery, due to the object momentarily sticking on the surface and being pushed again. For this reason we divided the measured sensor signals in 3 windows, and we extracted the mean, standard deviation and RMS value of each window as features. This leads to a smoothing on the signal jitter, reducing the uncertainty in quasi-static motion.

A. A note on cross-dataset learning

An obvious step when using two datasets is cross-testing, i.e. training on one dataset and testing on the other. The preferred option would be to train on the simulated dataset and test on the M-Dataset. An initial attempt resulted in model performance with error rates of 40-60%. The reasons for the poor performance are inherent to sim-to-real approaches. The frictional loads applied on the object from the surface can be controlled and measured in simulation, but they are inaccurate. On the other hand, in a real dataset frictional loads are physically accurate but difficult to measure. The control over the cross-dataset friction coefficients and the pressure distribution is also another parameter that makes sim-to-real testing difficult. The experimental conditions, such as pushing starting and ending points and sensor frequencies also affect the performance and are difficult to replicate. Our goal for this letter is to prove and test that the quasi-static pushing mechanics can be incorporated in a data-driven estimation model and not to provide a complete fit-for-all system. For that reason, we selected to leave the cross-testing analysis out of the letter's scope. Solving the cross-testing problem would open new frontiers in sim-to-real estimation and enable a system to be trained in simulation and deployed in the real world.

V. CONCLUSION

In this letter we presented a novel approach for estimating the inertial parameters of an object by pushing it. Our method addresses the shortcomings of previous works, by combining the accuracy of model-based estimation methods and the generalisation capability of data-driven methods. We designed experiments to show the estimation accuracy in both simulated and real conditions, over a range of inertial properties. The results were satisfactory, with our system achieving low means and relatively low variance among the predictions.

We used large datasets for the training purposes, and a future step would be to use learning methods that leverage big data, such as an LSTM or another deep learning framework,to achieve lower training and test errors. An extra step on this direction would be to use deep reinforcement learning methods for sim-to-real training, eliminating the need for cross training and allowing a robot to be trained on the simulated dataset and test its capability on various real datasets and testing samples.

Our previous research has showed how a robot can select among a set of grasps on an object by using the inertial parameters as an efficiency criterion [2][3]. This letter suggests a method for estimating the 2D inertial parameters with minimum interaction and relatively high fidelity. A next step would be to use the presented method's findings in order to find how a robot could estimate the full 3D inertial parameters of an object by simple interaction, and directly compute the manipulation criteria in [2][3].

An interesting case for our research would be the usage of our algorithm in closing the sim-to-real gap and deploy a robot in a real environment, where heavy objects need to be manipulated efficiently and accurate predictions are required. Finally, we aim to test our method with objects of variable mass distribution to further enrich our dataset and increase its capabilities.

REFERENCES

[1] V. Lippiello, "Grasp the Possibilities," *IEEE Robotics & Automation Magazine*, pp. 69–79, 2015.

[2] N. Mavrakis, A. M. Ghalamzan E., R. Stolkin, L. Baronti, M. Kopicki, and M. Castellani, "Analysis of the inertia and dynamics of grasped objects, for choosing optimal grasps to enable torque-efficient post-grasp manipulations," *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, pp. 171–178, 2016.

[3] N. Mavrakis, A. M. Ghalamzan E., and R. Stolkin, "Safe robotic grasping: Minimum impact-force grasp selection," *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-September, pp. 4034–4041, 2017.

[4] 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, p. 103374, 2020.

[5] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Rigid body load identification for manipulators," *Proceedings of IEEE Conference on Decision and Control*, vol. 24, no. December, 1985.

[6] D. Kubus, T. Kröger, and F. M. Wahl, "On-line estimation of inertial parameters using a recursive total least-squares approach," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3845–3852, 2008.

[7] B. Mirtich, "Fast and Accurate Computation of Polyhedral Mass Properties," *Journal of Graphics Tools*, vol. 1, pp. 31–50, 1996.

[8] M. Omid, M. Khojastehnazhand, and A. Tabatabaeefar, "Estimating volume and mass of citrus fruits by image processing technique," *Journal of Food Engineering*, vol. 100, no. 2, pp. 315–321, 2010.

[9] G. Vivek Venkatesh, S. M. Iqbal, A. Gopal, and D. Ganesan, "Estimation of Volume and Mass of Axi-Symmetric Fruits Using Image Processing Technique," *International Journal of Food Properties*, vol. 18, no. 3, pp. 608–626, 2015.

[10] T. Standley, O. Sener, D. Chen, and S. Savarese, "image2mass: Estimating the Mass of an Object from Its Image," *Proceedings of Conference on Robot Learning*, vol. 78, pp. 324–333, 2017.

[11] K. Fukuda and S. Tsujio, "Estimation of mass and center of mass of grasp less and shape-unknown object," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 2893–2898, 1999.

[12] S. Tanaka, T. Tanigawa, Y. Abe, M. Uejo, and H. T. Tanaka, "Active mass estimation with haptic vision," *Proceedings of International Conference on Pattern Recognition*, 2004.

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

[14] N. S. Methil and R. Mukherjee, "Pushing and steering wheelchairs using a holonomic mobile robot with a single arm," *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pp. 5781–5785, 2006.

[15] M. Artashes and D. Burschka, "Visual Estimation of Object Density Distribution through Observation of its Impulse Response," *Proceedings of the International Conference on Computer Vision Theory and Applications*, 2013.

[16] A. Franchi, A. Petitti, and A. Rizzo, "Decentralized parameter estimation and observation for cooperative mobile manipulation of an unknown load using noisy measurements," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 5517–5522, 2015.

[17] A. Franchi, A. Petitti, and A. Rizzo, "Distributed estimation of state and parameters in multiagent cooperative load manipulation," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 690–701, 2019.

[18] J. Wu, I. Yildirim, J. Lim, W. Freeman, and J. Tenenbaum, "Galileo : Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning," *Proceedings of Advances in Neural Information Processing Systems*, pp. 1–9, 2015.

[19] S. Zhu and A. Boularias, "A Physically-grounded and Data-efficient Approach to Motion Prediction using Black-box Optimization," *Proceedings of the Intuitive Physics Workshop at NIPS 2016*, pp. 3–6, 2016.

[20] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song, "DensePhysNet: Learning dense physical object representations via multi-step dynamic interactions," *Proceedings of Robotics: Science and Systems*, 2019.

[21] N. K. Kannabiran, I. Essa, and C. K. Liu, "Estimating mass distribution of articulated objects through physical interaction," *arXiv preprint arXiv:1907.03964*, 2019.

[22] 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," *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pp. 30–37, 2016.

[23] M. T. Mason, "Mechanics and Planning of Manipulator Pushing Operations," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[24] 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. 331–352, 1991.

[25] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction Part 2. Dynamics of motion," *Wear*, vol. 143, no. 2, pp. 331–352, 1991.

[26] K. Lynch, H. Maekawa, and K. Tanie, "Manipulation And Active Sensing By Pushing Using Tactile Feedback," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 416–421, 1991.

[27] K. T. Yu, J. Leonard, and A. Rodriguez, "Shape and pose recovery from planar pushing," *Procedings of IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 1208–1215, 2015.

[28] S. H. Lee and M. Cutkosky, "Fixture planning with friction," *Journal of Engineering for Industry*, vol. 113, no. 3, pp. 320–327, 1991.