Proceedings of Machine Learning Research vol 120:1–12, 2020

# Identifying Mechanical Models through Differentiable Simulations

**Changkyu Song** CS1080@RUTGERS.EDU
**Abdeslam Boularias** ABDESLAM.BOULARIAS@RUTGERS.EDU
*Computer Science Department, Rutgers, the State University of New Jersey, Piscataway, NJ, USA.*

## Abstract

This paper proposes a new method for manipulating unknown objects through a sequence of non-prehensile actions that displace an object from its initial configuration to a given goal configuration on a flat surface. The proposed method leverages recent progress in differentiable physics models to identify unknown mechanical properties of manipulated objects, such as inertia matrix, friction coefficients and external forces acting on the object. To this end, a recently proposed differentiable physics engine for two-dimensional objects is adopted in this work and extended to deal forces in the three-dimensional space. The proposed model identification technique analytically computes the gradient of the distance between forecasted poses of objects and their actual observed poses, and utilizes that gradient to search for values of the mechanical properties that reduce the reality gap. Experiments with real objects using a real robot to gather data show that the proposed approach can identify mechanical properties of heterogeneous objects on the fly.

## 1. Introduction

In robotics, nonprehensile manipulation can be more advantageous than the traditional pick-and-place approach when an object cannot be easily grasped by the robot Shome et al. (2019). For example, combined pushing and grasping actions have been shown to succeed where traditional grasp planners fail, and to work well under uncertainty by using the funneling effect of pushing Dogar and Srinivasa (2011). Environmental contact and compliant manipulation were also leveraged in peg-in-hole planning tasks Guan et al. (2018). Nonprehensile actions, such as pushing tabletop objects, have also been used for efficient rearrangement of clutter Dogar and Srinivasa (2012); King et al. (2015, 2016, 2017). The problem of pushing a single target object to a desired goal region was previously solved through model-free reinforcement learning Pinto et al. (2018); Yuan et al. (2018); Boularias et al. (2015), but this type of methods still requires colossal amounts of data and does not adapt rapidly to changes in the environment. We consider here only model-based approaches.

The mechanics of planar pushing was explored in the past Mason (1986), and large datasets of images of planar objects pushed by a robot on a flat surface were also recently presented N. Fazeli and Rodriguez (2017); M. Bauza and Rodriguez (2019). While this problem can be solved to a certain extent by using generic end-to-end machine learning tools such as neural networks M. Bauza and Rodriguez (2019), model identification methods that are explicitly derived from the equations of motion are generally more efficient. For instance, the recent work by Zhou et al. (2018) presented a simple and statistically efficient model identification procedure using a sum-of-squares convex relaxation, wherein friction loads of planar objects are identified from observed motions of objects. Other related works use physics engines as black-boxes, and global Bayesian optimization tools for inferring mechanical properties of objects, such as friction and mass Zhu et al. (2018b).

© 2020 C. Song & A. Boularias.

An alternative approach is to directly differentiate the simulation error with respect to the model of the object's parameters, and use standard gradient descent algorithms to search for optimal parameters. An advantage of this approach is the computational efficiency resulting from the guided search. Unfortunately, most popular physics engines do not natively provide the derivatives along with the 6D poses of the simulated rigid bodies Erez et al. (2015); DAR; Phy; Bul; ODE. The only way to differentiate them is to use finite differences, which is expensive computationally.

Figure 1: Robotic setup used in the experiments

In the present work, we consider the general problem of sliding an unknown object from an initial configuration to a target one while tracking a predefined path. We adopt the recent differentiable physics engine proposed in de Avila Belbute-Peres and Kolter (2017), and extend it to take into account frictional forces between pushed objects and their support surface. To account for non-uniform surface properties and mass distributions, the object is modeled as a large number of small objects that may have different material properties and that are attached to each other with fixed rigid joints. A simulation error function is given as the distance between the centers of the objects in simulated trajectories and the true ones. The gradient of the simulation error is then used to steer the search for the mass and friction coefficients.

## 2. Related Work

*Classical system identification* builds a dynamics model by minimizing the difference between the model's output signals and real-world response data for the same input signals Swevers et al. (1997); Ljung (1999). Parametric rigid body dynamics models have also been combined with non-parametric model learning for approximating the inverse dynamics of a robot Nguyen-Tuong and Peters (2010). *Model-based reinforcement learning* explicitly learns the unknown dynamics, often from scratch, and searches for an optimal policy accordingly Dogar et al. (2012); Lynch and Mason (1996); Scholz et al. (2014); Zhou et al. (2016); Abbeel et al. (2006). Particularly, *Gaussian Processes* have been widely used to model dynamical systems Deisenroth et al. (2011); Calandra et al. (2016); Marco et al. (2017); Bansal et al. (2017); Pautrat et al. (2017). Black-box, derivative-free, Covariance Matrix Adaptation algorithm (`CMA`) Hansen (2006) is also shown to be efficient at learning dynamical models Chatzilygeroudis et al. (2017). CMA was also used for automatically designing open-loop reference trajectories Tan et al. (2016), wherein trajectory optimization was performed in a physics simulator before real-world execution and real robot trajectories were used to fine-tune the simulator's parameters, the actual focus being the gains on the actuators. *Bayesian optimization* (`BO`) Shahriari et al. (2016) is a black-box approach that builds a probabilistic model of an objective function that does not have a known closed form. A GP is often used to model the unknown function. BO has been frequently used in robotics for direct policy optimization Pautrat et al. (2017), such as gait optimization Calandra et al. (2016). BO could also be used to optimize a policy while balancing the trade-off between simulation and real-world experiments Marco et al. (2017), or to learn a locally linear dynamics model Bansal et al. (2017). BO was also success-

fully used to identify mechanical models of simple tabletop objects from a few images Zhu et al. (2018a), and to search for robotic manipulation policies based on the identified models Zhu et al. (2018b). There has been a recent surge of interest in developing *natively differentiable physics engines*. For example, Degrave et al. (2016) used the Theano framework Al-Rfou et al. (2016) to develop a physics engine that can be used to differentiate control parameters in robotics applications. The same framework can be altered to differentiate model parameters instead. A combination of a learned and a differentiable simulator was used to predict action effects on planar objects Kloss et al. (2017). A differentiable framework for integrating fluid dynamics with deep networks was also used to learn fluid parameters from data, perform liquid control tasks, and learn policies to manipulate liquids Schenck and Fox (2018). Differentiable physics simulations were also used for manipulation planning and tool use Toussaint et al. (2018). Recently, it has been observed that a standard physical simulation, which iterates solving the Linear Complementary Problem (LCP), is also differentiable and can be implemented in PyTorch de Avila Belbute-Peres and Kolter (2017). In Mordatch et al. (2012), a differentiable contact model was used to allow for optimization of several locomotion and manipulation tasks.

## 3. Problem Setup and Notation

Parameters of material properties of objects, represented as a vector $\theta \in \Theta$ of dimension $D$, are given as input to the physics engine. For example, $\theta_{[0]}$ represents the object's density, while $\theta_{[1]}$ is the coefficient of friction between the object and a support surface. A real-world trajectory $\tau^*$ is a state-action sequence $(x_0, u_0, \ldots, x_{T-1}, u_{T-1}, x_T)$, where $x_t$ is the state of the pushed object, and $u_t$ is the action (force) applied at time $t$. In this work, we focus on the problem of pushing an unknown rigid object. The object is approximated as a finite set of elements. The object is thus composed of several connected cells $1, 2, \ldots, k$. Each cell $i$ has its own mechanical properties that can be different from other cells. State $x_t$ is a vector in $[SE(2)]^k$ corresponding to translation and rotation in the plane for each of the $k$ cells. A corresponding simulated trajectory $\tau$ for assumed physical parameters $\theta$ is obtained by starting at an initial state $\hat{x}_0$, which is set to be the same as the initial state of the corresponding real trajectory, i.e., $\hat{x}_0 = x_0$, and applying the same control sequence $(u_0, u_1, \ldots, u_{T-1})$. Thus, the simulated trajectory $\tau$ results in a state-action sequence $(\hat{x}_0, \hat{v}_0, u_0, \hat{x}_1, \hat{v}_1, u_1, \ldots, \hat{x}_{T-1}, \hat{v}_{T-1}, u_{T-1}, \hat{x}_T)$, where $\hat{x}_{t+1} = \hat{x}_t + \hat{v}_t dt$ is the next state predicted by the physics engine using the model $\theta$. Predicted velocity $\hat{v}_t$ is a vector in $[SE(2)]^k$ corresponding to translation and angular velocities in the plane for each of the $k$ cells. Predicted velocity $\hat{v}_{t+1}$ is given by $\hat{v}_{t+1} = V(\hat{x}_t, \hat{v}_t, u_t, \theta), \hat{v}_0 = \mathbf{0}$. The goal is to identify model parameters $\theta^*$ that result in simulated state outcomes $\hat{x}_{t+1}$ that are as close as possible to the real observed states $x_{t+1}$. In other words, the objective is to solve the following optimization problem:

$$\theta^* = \arg\min_{\theta \in \Theta} loss(\theta) \stackrel{def}{=} \sum_{t=0}^{T-2} \|x_{t+2} - \big(\hat{x}_{t+1} + V(\hat{x}_t, \hat{v}_t, u_t, \theta)dt\big)\|_2. \tag{1}$$

In the following, we explain how velocity function $V$ is computed.

$$\begin{bmatrix}0\\0\\a\\\rho\\\xi\end{bmatrix} - \begin{bmatrix}\mathcal{M} & -\mathcal{J}_e(x_t) & -\mathcal{J}_c(x_t) & -\mathcal{J}_f & 0\\ \mathcal{J}_e(x_t) & 0 & 0 & 0 & 0\\ \mathcal{J}_c(x_t) & 0 & 0 & 0 & 0\\ \mathcal{J}_f & 0 & 0 & 0 & E\\ 0 & 0 & \mu_f & -E^T & 0\end{bmatrix}\begin{bmatrix}v_{t+dt}\\\lambda_e\\\lambda_c\\\lambda_f\\\eta\end{bmatrix} = \begin{bmatrix}\mathcal{M}v_t + dt\mu_t\\0\\c\\0\\0\end{bmatrix} \text{ s.t. } \begin{bmatrix}a\\\rho\\\xi\end{bmatrix} \geq 0, \begin{bmatrix}\lambda_c\\\lambda_f\\\eta\end{bmatrix} \geq 0, \begin{bmatrix}a\\\rho\\\psi\end{bmatrix}\begin{bmatrix}\lambda_c\\\lambda_f\\\eta\end{bmatrix} = 0$$

Figure 2: Equations of Motion

## 4. Foundations

We adopt here the formulation presented by de Avila Belbute-Peres and Kolter (2017), and we extend it to include frictional forces between a pushed object and a support surface. The transition function is given as $x_{t+1} = x_t + v_t dt$ where $dt$ is the duration of a constant short time-step. Velocity $v_t$ is a function of pushing force $\mu_t$ and mechanical parameters $\theta$. To find $v_{t+dt}$, we solve the system of equations of motion given in Figure 2, where $x_t$ and $v_t$ are given as inputs, $[a, \rho, \xi]$ are slack variables, $[v_{t+dt}, \lambda_e, \lambda_c, \lambda_f, \eta]$ are unknowns, and $[\mathcal{M}, \mu_f] \overset{def}{=} \theta$ are the mechanical properties of the manipulated object that are hypothesized. $\mathcal{M}$ is a diagonal mass matrix, where the diagonal is given as $[\mathcal{I}_1, \mathcal{M}_1, \mathcal{M}_1, \mathcal{I}_2, \mathcal{M}_2, \mathcal{M}_2, \ldots, \mathcal{I}_k, \mathcal{M}_k, \mathcal{M}_k]$, where $\mathcal{I}_i$ is the moment of inertia of the $i^{th}$ cell of the object, and $\mathcal{M}_i$ is its mass. $\mu_F$ is a diagonal matrix containing the magnitudes of the frictional forces exerted on each cell of the pushed object in the direction opposite to its motion.

$\mathcal{J}_e(x_t)$ is a global Jacobian matrix listing all constraints related to the joints of the object. These constraints ensure that the different cells of the object move together with the same velocity, since the object is assumed to be rigid. $\mathcal{J}_c(x_t)$ is a matrix containing constraints related to contacts between the cells of the object. These constraints enforce that rigid cells of the object do not interpenetrate. $c$ is a vector that depends on the velocities at the contact points and on the combined restitution parameter of the cells. More detailed descriptions of $\mathcal{J}_e(x_t)$ and $\mathcal{J}_c(x_t)$ can be found in the supplementary material of Belbute-Peres et al. (2018).

$\mathcal{J}_f$ is a Jacobian matrix related to the frictional forces, and it is particularly important for the objectives of this work. We will return to $\mathcal{J}_f$ in Section 5. To present the solution more concisely, the following terms are introduced.

$$\alpha = -v_{t+dt}, \beta = \lambda_e, A = \mathcal{J}_e(x_t), q = -\mathcal{M}v_t - dt\mu_t$$

$$\gamma = \begin{bmatrix}\lambda_c\\\lambda_f\\\eta\end{bmatrix}, s = \begin{bmatrix}a\\\rho\\\xi\end{bmatrix}, m = \begin{bmatrix}c\\0\\0\end{bmatrix}, s = \begin{bmatrix}a\\\rho\\\xi\end{bmatrix}, G = \begin{bmatrix}\mathcal{J}_e(x_t) & 0\\ \mathcal{J}_f & 0\\ 0 & 0\end{bmatrix}, \mathcal{F} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & E\\ \mu_f & -E^T & 0\end{bmatrix}$$

The linear complementary problem (LCP) becomes

$$\begin{bmatrix}0\\s\\0\end{bmatrix} + \begin{bmatrix}\mathcal{M} & G^T & A^T\\ G & \mathcal{F} & 0\\ A & 0 & 0\end{bmatrix}\begin{bmatrix}\alpha\\\beta\\\gamma\end{bmatrix} = \begin{bmatrix}-q\\m\\0\end{bmatrix} \tag{2}$$

subject to $s \geq \mathbf{0}, \gamma \geq \mathbf{0}, s^T\gamma = 0$.

The solution is obtained, after an initialization step, by iteratively minimizing the residuals from the equation above through variables $\alpha, \beta, \gamma$ and $s$. The solution is obtained by utilizing the convex optimizer of Mattingley and Boyd (2012).

## 5. Friction Model

We describe the Jacobian matrix $\mathcal{J}_f$ related to the Coulomb frictional forces between the object's cells and the support surface, and the corresponding constraints. This model is based on the one derived in Cline (2002).

$$\mathcal{D} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \mathcal{J}_f = \begin{bmatrix} \mathcal{D} & 0 & \dots & 0 \\ 0 & \mathcal{D} & \dots & 0 \\ \vdots & \vdots & \dots & 0 \\ 0 & 0 & \dots & \mathcal{D} \end{bmatrix}, \mu_f = \begin{bmatrix} \mu_{f_1} & 0 & \dots & 0 \\ 0 & \mu_{f_2} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \mu_{f_k} \end{bmatrix}, E = \begin{bmatrix} \zeta & 0 & \dots & 0 \\ 0 & \zeta & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \zeta \end{bmatrix}$$

$\zeta = [1, 1, 1, 1]^T$, $\mathcal{J}_f$ and $E$ are both a $k \times (4k)$ matrix where $k$ is the number of cells in the object. $\mu_{f_i}$ is the unknow friction coefficient of cell $i$ with the support surface. In the original model of Cline (2002), matrix $\mathcal{D}$ was defined as $\mathcal{D} = \begin{bmatrix} (p_{contact} \times d) & d \\ (p_{contact} \times -d) & -d \end{bmatrix}$, where $d$ is a vector pointing to the tangential to a contact, and $-d$ is a vector pointing to the opposite direction. $p_{contact}$ is the contact point on the object. In our model, we consider only pushing actions that slide objects forward without rolling them over. We thus eliminate the first column of $\mathcal{D}$ which are multiplied by the angular velocities and replace it with a zero column. The friction terms have complementary constraints that are related to the contact points. We will omit the derivations here (see Cline (2002)), but interactions between the objects and the support surface give rise to the following constraints, $\mu_f - E^T\lambda_f \geq 0, \mathcal{J}_f v_{t+dt} + \gamma E \geq 0, \rho^T \lambda_f = 0, \xi^T \eta = 0$.

## 6. Proposed Algorithm

To obtain parameters $\theta^* = [\mathcal{M}^*, \mu_f^*]$, a gradient descent on loss function in Equation 1 is performed, wherein the gradients are computed analytically. A first approach to compute the gradient is to use the *Autograd* library for automatic derivation in Python. We propose here a second simpler and faster approach that exploits three mild assumptions: 1) the manipulated object is rigid, 2) the contact point between the robot's end-effector and the object remains constant during the pushing action, and 3) collision between the robot's end-effector and the object is perfectly inelastic with a zero restitution coefficient. That is, the end-effector remains in contact with the object during the pushing action. The relatively low velocity of the end-effector (around 0.2 m/s) ensures the inelastic nature of the collision. The equation of motion can be written as:

$$\mathcal{M}v_{t+dt} - \mathcal{J}_e(x_t)\lambda_e - \mathcal{J}_c(x_t)\lambda_c - \mu_f\lambda_c = \mathcal{M}v_t + dt\mu_t. \tag{3}$$

Thus, $V(\hat{x}_t, \hat{v}_t, u_t, \theta) = v_{t+dt} = \mathcal{M}^{-1}\big(\mathcal{J}_e(x_t)\lambda_e + \mathcal{J}_c(x_t)\lambda_c + \mu_f\lambda_c + \mathcal{M}v_t + dt\mu_t\big)$. Inverse mass matrix $\mathcal{M}^{-1}$ exists because $\mathcal{M}$ is a diagonal full-rank matrix. The gradients of predicted velocity with respect to $\mu_f$ is give as: $\nabla_{\mu_f} V(\hat{x}_t, \hat{v}_t, u_t, \theta) = \mathcal{M}^{-1}\lambda_c$, because $\nabla_{\mu_f}\mathcal{J}_e(x_t)\lambda_e = \nabla_{\mu_f}\mathcal{J}_c(x_t)\lambda_c = 0$ from assumptions 1-3. The gradient of the loss in Equation 1 is given by $\nabla_{\mu_f} loss(\theta) = \sum_{t=0}^{T-2} \nabla_{\mu_f} \|x_{t+2} - \big(\hat{x}_{t+1} + V(\hat{x}_t, \hat{v}_t, u_t, \theta)dt\big)\|_2 = \sum_{t=0}^{T-2} \Big(x_{t+2} - \big(\hat{x}_{t+1} + V(\hat{x}_t, \hat{v}_t, u_t, \theta)dt\big)\Big)\mathcal{M}^{-1}\lambda_c dt = C_f \sum_{t=0}^{T-2} \Big(x_{t+2} - \big(\hat{x}_{t+1} + \underbrace{V(\hat{x}_t, \hat{v}_t, u_t, \theta)}_{\text{forward simulation}}\big)\Big)$ where $C_f$ is a constant diagonal matrix. The real value of $C_f$ is of little importance as it will be absorbed by the learning rate of the gradient descent algorithm. A similar derivation for $\mathcal{M}$ gives $\nabla_{\mathcal{M}} loss(\theta) = C_m \sum_{t=0}^{T-2} \Big(x_{t+2} - \big(\hat{x}_{t+1} + V(\hat{x}_t, \hat{v}_t, u_t, \theta)\big)\Big)$, with $C_M = \alpha C_f^{-\frac{1}{2}}$ and $\alpha$ is a constant factor. Thus,

we use update rates $\alpha_{\text{rate}}$ and $\sqrt{\alpha_{\text{rate}}}$ for frictional forces magnitudes and mass matrix respectively, as as shown in Algorithm 1.

**Input:** Real-world trajectory data $\tau^* = \{(x_t, \mu_t, x_{t+1})\}$ for $t = 0, \ldots, T-1$, wherein $x_t$ is a vector in $[SE(2)]^k$ corresponding to translation and rotation in the plane for each of the $k$ cells of a pushed unknown object, and $\mu_t$ is a force described by the contact point between the end-effector and one of the object's cells in addition to the pushing direction. Predefined learning rate $\alpha_{\text{rate}}$, and loss threshold $\epsilon$.
**Output:** Parameters vector $\theta = [\mathcal{M}, \mu_f]$
Initialize $\mu_f$ randomly;
**repeat**
  $loss \leftarrow 0$;
  **for** $t \in \{0, T-2\}$ **do**
    Simulate $\{(\hat{x}_{t+1}, \mu_{t+1})\}$ by solving the LCP in Equation 2 with parameters $\theta$ (or by using an off-the shelf physics engine) and get the predicted next state $\hat{x}_{t+2} = \hat{x}_{t+1} + V(\hat{x}_t, \hat{v}_t, u_t, \theta)$;
    $loss \leftarrow loss + \|\hat{x}_{t+2} - x_{t+2}\|_2$;
    $\mu_f \leftarrow \mu_f + \alpha_{\text{rate}}(\hat{x}_{t+2} - x_{t+2})$;
    $\mathcal{M} \leftarrow \mathcal{M} + \sqrt{\alpha_{\text{rate}}}(\hat{x}_{t+2} - x_{t+2})$;
  **end**
**until** $loss \leq \epsilon$;

**Algorithm 1:** Learning Mechanical Models with Differentiable Physics Simulations

## 7. Evaluation

We report here the results of our experiments for evaluating the proposed method.

### 7.1. Baselines

The compared baselines are *random search*, *finite differences gradient*, *automatic differentiation with Autograd*, and *weighted sampling*. The weighted sampling search generates random values uniformly in the first iteration, and then iteratively generates normally distributed random values around the best parameter obtained in the previous iteration. The standard deviation of the random values is gradually reduced over time, to focus the search on the most promising region.

### 7.2. Experimental Setup

The experiments are performed on both simulated and real robot and objects. A rigid object is set on a table-top. The robot's end effector is moved randomly to collide with the object and push it forward. The initial and final poses of the object are recorded. The methods discussed above are used to estimate the object's mass and frictional forces that are distributed over its cells. Since the ground-truth values of mass and friction are unknown, the identified models are evaluated in terms of the accuracy in the predicted pose of each cell, using a set of test data. Five different objects are used in our experiments: a hammer, a ranch, a crimp, a toolbox, and a book. A hammer has an unbalanced mass distribution because the iron head is much heavier than the wooden handle. A crimp has high frictional forces on its heavy iron head and stiff handle. A ranch is composed of the same material, however, its handle in the middle (main body) floats and does not touch the table, because of the elevated height of its side parts. Therefore, there are zero frictional forces on the handle. Finally, an open book that has a different number of pages on the left and right sides also

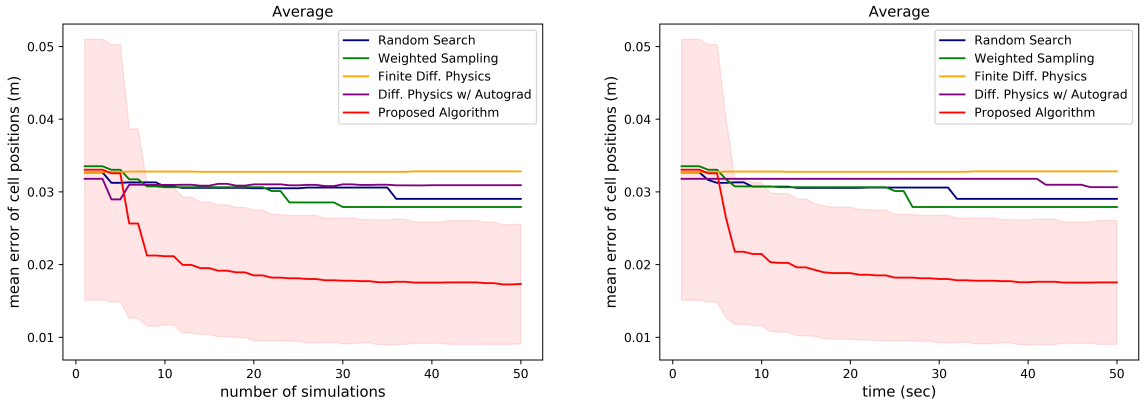



Figure 3: Average predicted cell position error (in meters) over all objects and cells for each object as a function of the number of simulations (left) and the computation time (right).

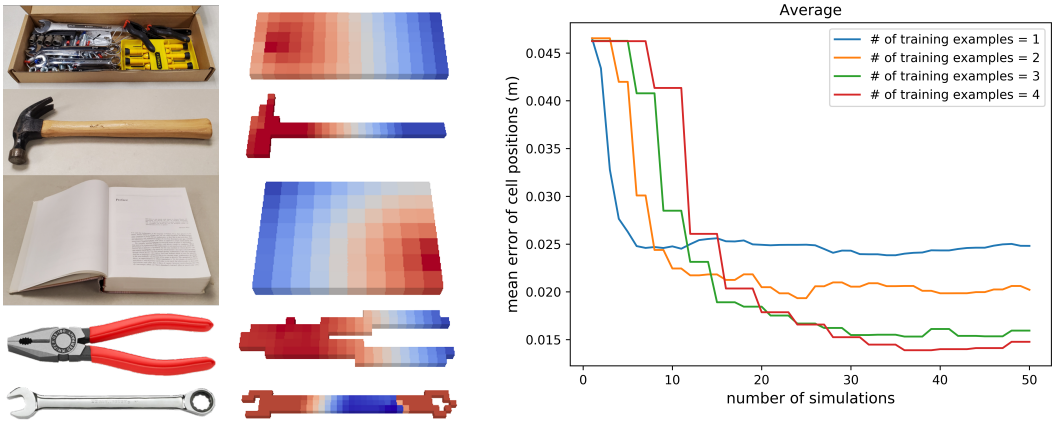
has an unbalanced mass-friction distribution, resulting in rotations when pushed. A toolbox also can have a various mass distribution depending on how the tools are arranged inside.

Note that our method does not assume that the full shape of the object is known, it uses only the observed top part and projects it down on the table to build a complete 3D geometric model. The geometric model is generally wrong because the object is seldom flat, but the parts that do not actually touch the table end up having nearly zero frictions in the identified model. Thus, identified low friction forces compensate for wrongly presumed surfaces in the occluded bottom part of the object, and the predicted motion of the object is accurate despite using inaccurate geometries.

In the simulation experiments, we simulated four random actions on each of the following objects for collecting training data: hammer, crimp, and ranch. We use the physics engine *Bullet* for that purpose. The results are averaged over ten independent experiments, with a different ground-truth model of the object used in each experiment to generate data. The identified mass and friction models are evaluated by measuring the accuracy of the predicted motions on a test set of 12 different random pushing actions. In the real robot setup, we used a Kuka robotic arm and Robotiq 3-finger hand to apply the pushing actions, and recorded the initial and final object poses using a depth-sensing camera, as shown as Figure 1. The training data set contains only two random pushing actions, while the test data set contains five random pushing actions. The goal is to show how the robot can identify models of objects with a very small number of manipulation actions. The number of cells per object varies from 70 to 100 depending on the size of the object.

### 7.3. Results

Figure 4 (a) shows identified mass-friction distributions, given as the product of the mass and the friction coefficients of each cell. The mass-friction distributions of the first three objects are estimated in the real robot setup, and the last two are estimated in the simulation setup. The toolbox contains heavy tools on the left side (ranches, bolts and nuts), while relatively light tools like plastic screw drivers and cramps are placed on the right side. The proposed method was able to predict the unbalanced mass distribution of the box while it was covered. Likewise, the heavier iron head of the hammer and its light wooden handle are successfully estimated as well as the thicker and thinner sides of the book. The proposed method successfully estimated the heavier part of the crimp, and the floating part of the ranch was simulated by much lighter friction values in the middle.




(a) (b)

Figure 4: (a) Learned friction×mass distributions. Red color means higher mass×friction value while blue color means lower mass×friction value. (b) Predicted cell position error with different numbers of training examples and simulations (gradient-descent steps).

Figure 3 shows the difference between the predicted cell positions and the ground truth as a function of the number simulations used in the parameter estimations. The results first demonstrate that global optimization methods (random and weighted sampling) suffer from the curse of dimensionality due to the combinatorial explosion in the number of possible parameters for all cells. The results also demonstrate that the proposed method can estimate the parameters within a small number of simulations and a short computation time. The proposed algorithm was able to estimate the parameters with under $1.5cm$ average cell position error within 30 seconds. Surprisingly, the differential physics engine with *Autograd* requires a significantly longer computation time as the number of cells increases, which is a critical in practice. The finite differences approach also failed to converge to an accurate model due to the high computational cost of the gradient computation, as well as the sensitivity of the computed gradients to the choice the grid size.

Finally, Figure 4 (b) shows how the number of training actions improves the accuracy of the learned model. Increasing the number of training actions allows the robot to uncover properties of different parts of the object more accurately. Using a larger number of training actions slows down the convergence of the gradient-descent algorithm, but improves the accuracy of the learned model.

## 8. Conclusion

To identify friction and mass distributions of unknown objects pushed by a robot, we proposed a new method that consists in dividing an object into a large number of connected cells, with each cell having different mechanical properties. We adopted a differentiable physics engine that was recently proposed to simulate contact interactions between 2-dimensional objects, and we extended it to deal with frictional forces occurring on table-top 3D objects. In addition to the automatic derivation of the engine, based on *Autograd*, we presented a simple gradient-descent algorithm that exploits weak assumptions about the object and the collision to simplify the form of the gradient of reality-gap loss function with respect to the object's parameters. The proposed algorithm was tested in simulation and with real objects, and shown to be efficient in identifying models of objects with non-uniform mass-friction distributions.

**Acknowledgments** This work was supported by NSF awards 1734492, 1723869 and 1846043.

## References

Bullet physics engine. [Online]. Available: www.bulletphysics.org.

DART physics egnine. [Online]. Available: http://dartsim.github.io.

Open dynamics engine. [Online]. Available: http://ode.org.

PhysX physics engine. [Online]. Available: www.geforce.com/hardware/technology/physx.

Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *Proc. of ICML*. ACM, 2006.

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermüller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul F. Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron C. Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Melanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian J. Goodfellow, Matthew Graham, Çaglar Gülçehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrançois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Joseph Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph P. Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688, 2016.

Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomlin. Goal-driven dynamics learning via bayesian optimization. In *IEEE CDC*, 2017.

Filipe de A. Belbute-Peres, Kevin A. Smith, Kelsey R. Allen, Joshua B. Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 7178–7189, USA, 2018. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=3327757.3327820.

Abdeslam Boularias, James Andrew Bagnell, and Anthony Stentz. Learning to Manipulate Unknown Objects in Clutter by Reinforcement. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc P. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 76 (1):5–23, 2016. ISSN 1573-7470.

Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. Black-Box Data-efficient Policy Search for Robotics. In *IROS*, September 2017. URL https://hal.inria.fr/hal-01576683.

Michael Bradley Cline. *Rigid body simulation with contact and constraints*. PhD thesis, University of British Columbia, 2002.

Filipe de Avila Belbute-Peres and Zico Kolter. A modular differentiable rigid body physics engine. In *Deep Reinforcement Learning Symposium, NIPS*, 2017.

Jonas Degrave, Michiel Hermans, Joni Dambre, and Francis Wyffels. A differentiable physics engine for deep learning in robotics. *CoRR*, abs/1611.01652, 2016.

M. Deisenroth, C. Rasmussen, and D. Fox. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *R:SS*, 2011.

Mehmet Dogar, Kaijen Hsiao, Matei Ciocarlie, and Siddhartha Srinivasa. Physics-Based Grasp Planning Through Clutter. In *R:SS*, July 2012.

Mehmet R. Dogar and Siddhartha S. Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33(3):217–236, Oct 2012.

Mehmet Remzi Dogar and Siddhartha S. Srinivasa. A framework for push-grasping in clutter. In *Robotics: Science and Systems*, 2011.

Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ODE and physx. In *IEEE ICRA*, 2015.

Charlie Guan, William Vega-Brown, and Nicholas Roy. Efficient planning for near-optimal compliant manipulation leveraging environmental contact. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 215–222, 2018. doi: 10.1109/ICRA.2018.8462696. URL https://doi.org/10.1109/ICRA.2018.8462696.

Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

J. E. King, V. Ranganeni, and S. S. Srinivasa. Unobservable monte carlo planning for nonprehensile rearrangement tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4681–4688, May 2017. doi: 10.1109/ICRA.2017.7989544.

Jennifer E. King, Joshua A. Haustein, Siddhartha S. Srinivasa, and Tamim Asfour. Nonprehensile whole arm rearrangement planning on physics manifolds. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 2508–2515. IEEE, 2015. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139535. URL https://doi.org/10.1109/ICRA.2015.7139535.

Jennifer E. King, Marco Cognetti, and Siddhartha S. Srinivasa. Rearrangement planning using object-centric and robot-centric action spaces. In *2016 IEEE International Conference on Robotics and Automation, ICRA, Stockholm, Sweden, May 16-21, 2016*, pages 3940–3947, 2016.

Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects. *CoRR*, abs/1710.04102, 2017.

Lennart Ljung, editor. *System Identification (2Nd Ed.): Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0-13-656695-2.

K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, control- lability, and planning. *International Journal of Robotics Research*, 18, 1996.

Y. Lin T. Lozano-Perez L. Kaelbling P. Isola M. Bauza, F. Alet and A. Rodriguez. Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgbd images. In *IROS*, 2019.

Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P. Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization. In *ICRA*, pages 1557–1563, 2017.

Matthew T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

J. Mattingley and S. Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 12(1):1–27, 2012.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4):43:1–43:8, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185539. URL http://doi.acm.org/10.1145/2185520.2185539.

R. Tedrake N. Fazeli, R. Kolbert and A. Rodriguez. Parameter and contact force estimation of planar rigid-bodies undergoing frictional contact. *IJRR*, 2017.

Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *ICRA*, pages 2677–2682, 2010.

Rémi Pautrat, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. Bayesian optimization with automatic prior selection for data-efficient direct policy search. *arXiv preprint arXiv:1709.06919*, 2017.

Lerrel Pinto, Aditya Mandalika, Brian Hou, and Siddhartha Srinivasa. Sample-efficient learning of nonprehensile manipulation policies via physics-based informed state distributions. *CoRR*, abs/1810.10654, 2018. URL http://arxiv.org/abs/1810.10654.

Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. *CoRR*, abs/1806.06094, 2018.

Jonathan Scholz, Martin Levihn, Charles L. Isbell, and David Wingate. A Physics-Based Model Prior for Object-Oriented MDPs. In *ICML*, 2014.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.

Rahul Shome, Wei Neo Tang, Changkyu Song, Chaitanya Mitash, Chris Kourtev, Jingjin Yu, Abdeslam Boularias, and Kostas Bekris. Towards robust product packing with a minimalistic end-effector. In *IEEE International Conference on Robotics and Automation, ICRA 2019, Montreal, Canada*, 2019.

Jan Swevers, Chris Ganseman, D Bilgin Tukel, Joris De Schutter, and Hendrik Van Brussel. Optimal robot excitation and identification. *IEEE TRO-A*, 13(5):730–740, 1997.

Jie Tan, Zhaoming Xie, Byron Boots, and C Karen Liu. Simulation-based design of dynamic controllers for humanoid balancing. In *IROS*. IEEE, 2016.

Marc Toussaint, Kelsey R Allen, Kevin A Smith, and Josh B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Proc. of Robotics: Science and Systems (R:SS 2018)*, 2018.

Weihao Yuan, Johannes A. Stork, Danica Kragic, Michael Yu Wang, and Kaiyu Hang. Rearrangement with nonprehensile manipulation using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 270–277, 2018. doi: 10.1109/ICRA.2018.8462863. URL https://doi.org/10.1109/ICRA.2018.8462863.

Jiaji Zhou, Robert Paolini, J. Andrew Bagnell, and Matthew T. Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *ICRA*, pages 372–377, 2016.

Jiaji Zhou, Matthew T Mason, Robert Paolini, and Drew Bagnell. A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation. *The International Journal of Robotics Research*, 37(2-3):249–265, 2018.

Shaojun Zhu, Andrew Kimmel, Kostas Bekris, and Abdeslam Boularias. Fast Model Identification via Physics Engines for Data-Efficient Policy Search. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018a.

Shaojun Zhu, David Surovik, Kostas Bekris, and Abdeslam Boularias. Efficient Model Identification for Tensegrity Locomotion . In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018b.