A Physically-grounded and Data-efficient Approach to Motion Prediction using Black-box Optimization

Shaojun Zhu and Abdeslam Boularias
Department of Computer Science
Rutgers University
`{shaojun.zhu, abdeslam.boularias}@cs.rutgers.edu`

Abstract

In this paper, we introduce a practical, data-efficient approach for identifying sliding models of objects. In the proposed approach, a robot randomly pokes an unknown object and observes how the object moves under different force magnitudes and directions. Using a physics engine, the robot seeks to identify the inertial and friction parameters of the object by simulating its motion under different values of the parameters and identifying those that result in a simulation that matches observed motions. We describe the proposed method here and report some preliminary experimental results, using a real robot, which are encouraging.

1 Introduction

Predicting object motion under physical interaction is one of the key challenges in applying robotics in everyday life. Recent advances in Deep Neural Networks(DNNs) motivated several works to apply DNNs for such problems. However, these approaches usually require large amounts of training data, from either simulation or real-world roll-outs.

We are proposing a data-efficient approach for motion prediction by utilizing a physics engine and learning the physical parameters through black-box Bayesian optimization. We are motivated by recent work[6, 7] that uses physics engines generating data for physical stability prediction.

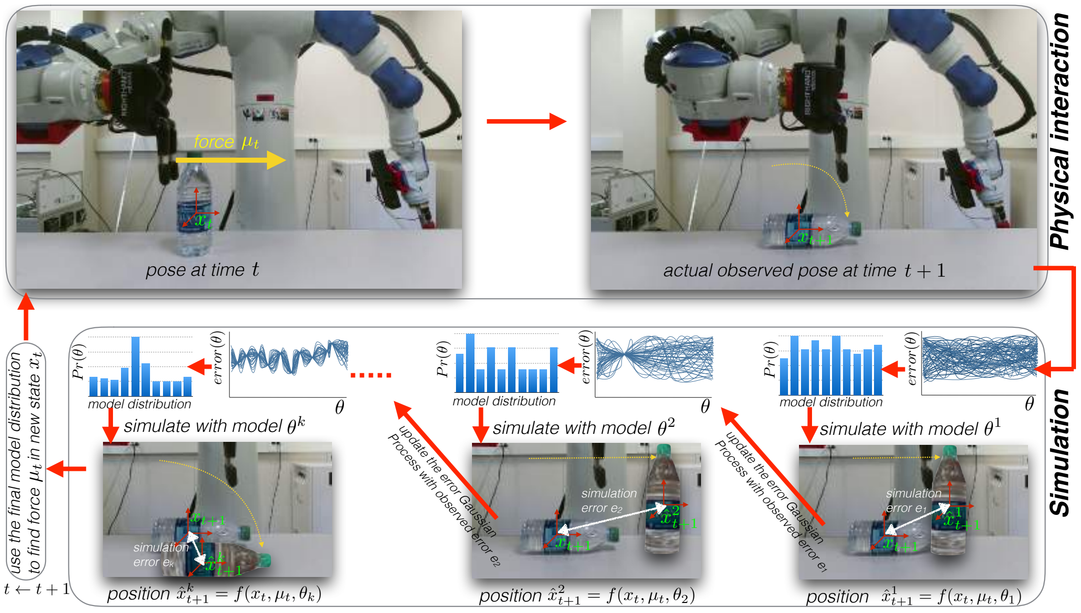
Specifically, we are interested in predicting the motion of an object under the action of a robotic hand. First, we use a real robot to perform some random poking action with an object on a tabletop[2]. We record both the initial and final configurations of the object and the hand. Instead of learning the object's motion explicitly, we use a Bayesian optimization technique to identify related physical parameters like mass and friction through the physics engine simulation.

To predict the motion of the object under a new action, we use the learned parameters to simulate the action in the physics engine and use the result of the simulation as our prediction.

2 Proposed Approach

To solve the problem of modeling mechanical properties of objects, we propose an online learning approach to identify mass and sliding models of objects as a Bayesian optimization problem. The goal is to allow the robot to use predefined models of objects, in the form of prior distributions, and to improve the accuracy of these models on the fly by interacting with the objects. The learning process should be in real time because it takes place simultaneously with the physical interaction.

Figure 1 shows an overview of the proposed approach. The first step consists in using a pre-trained object detector to detect the different objects present in the scene. We are using a SIFT-based tracker to detect objects and estimate their poses, by mapping them to a knowledge base of pre-existing 3D



force μ_t
x_t
pose at time t
actual observed pose at time $t+1$
Physical interaction
Simulation
$Pr(\theta)$
$error(\theta)$
θ
model distribution
simulate with model θ^k
simulate with model θ^2
simulate with model θ^1
update the error Gaussian Process with observed error e_2
update the error Gaussian Process with observed error e_1
simulation error e_k
simulation error e_2
simulation error e_1
position $\hat{x}^k_{t+1} = f(x_t, \mu_t, \theta_k)$
position $\hat{x}^2_{t+1} = f(x_t, \mu_t, \theta_2)$
position $\hat{x}^1_{t+1} = f(x_t, \mu_t, \theta_1)$
use the final model distribution to find force μ_t in new state x_t
$t \leftarrow t+1$

Figure 1: Overview of the proposed approach for learning object models with a physics engine

mesh models. We propose to augment the 3D mesh models with mechanical properties of objects. The mechanical properties correspond to the mass and the static and kinetic friction coefficient of each subpart of a given object. These properties are represented as a d-dimensional vector θ. A prior distribution P_0 on θ is used instead of a single value of θ, because different instances of the same category usually have different mechanical properties.

The online learning algorithm takes as input a prior distribution P_t on model parameters θ. Distribution P_t is calculated based on the initial distribution P_0 and a sequence of observations $(x_0, \mu_0, x_1, \mu_1, \dots, x_{t-1}, \mu_{t-1}, x_t)$, wherein x_t is the 6D pose (position and orientation) of the manipulated object at time t and μ_t is a vector describing a force applied by the robot's fingertip on the object at time t. Applying a force μ_t results in changing the object's pose from x_t to x_{t+1}.

Given a prior distribution P_t and a new observation $(x_t, \mu_{t+1}, x_{t+1})$, we use a physics engine to estimate a posterior distribution P_{t+1} on the model parameters θ. We are currently using the Bullet physics engine in our preliminary experiments [1]. The posterior distribution P_{t+1} is obtained by simulating the effect of force μ_{t+1} on the object under various values of parameters θ and observing the resulting positions $\hat{x}_{t+1}$. The goal is to identify the model parameters that make the outcome $\hat{x}_{t+1}$ of the simulation as close as possible to the actual observed outcome x_{t+1}. In other terms, we solve the following black-box optimization problem:

$$\theta^* = \arg\min_{\theta} E(\theta) \overset{def}{=} \|x_{t+1} - f(x_t, \mu_t, \theta)\|_2,$$

wherein x_t and x_{t+1} are the observed poses of the object at times t and $t+1$, μ_t is the force that moved the object from x_t to x_{t+1}, and $f(x_t, \mu_t, \theta) = \hat{x}_{t+1}$, the simulated pose at time $t+1$ after applying force μ_t in pose x_t.

The model parameters θ can be limited to a discrete set, i.e. $\theta \in \{\theta^1, \theta^2, \dots, \theta^n\} \overset{def}{=} \Theta$. A naive approach to solving this problem consists in systematically simulating all the parameters θ^i in Θ, simulating the effect of force μ_t on the object with parameters θ^i, and comparing the predicted pose $f(x_t, \mu_t, \theta^i)$ to the actual pose x_{t+1}. However, this would be inefficient because the size of Θ should be relatively high given that the dimension d of the parameter space is typically high. Each individual simulation is also computationally expensive. Therefore, we need to minimize the number of simulations while searching for the optimal parameters. Moreover, the optimization problem above is ill-posed, as is the case in all inverse problems. In other terms, there are multiple model parameters that can explain an observed movement of an object. Therefore, our algorithm returns a posterior distribution P_{t+1} on the set of possible parameters Θ, instead of returning a single answer.

We formulate this problem in the Bayesian optimization framework, and we use the Entropy Search technique originally presented in [5]. We specifically use a greedy-search variant that we proposed in [3] and that is more efficient computationally. This method is explained below.

To solve the optimization problem above, we need to learn the error function E from a minimum number of simulations, using a sequence of parameters $\theta_1, \theta_t, \dots, \theta_k \in \Theta$. To choose these parameters efficiently, we maintain a belief about the actual error function. This belief is a probability measure $p(E)$ over the space of all functions $E : \mathbb{R}^d \rightarrow \mathbb{R}$. We use a Gaussian Process (GP) to represent the belief p, which is sequentially updated using the errors $E(\theta_i)$ computed from simulation using model parameters θ_i. We refer the reader to [9] for more details on how Gaussian processes are updated. Belief p is initialized at each time t using prior P_t, the model distribution from the previous time-step.

After simulating the object's motion with different model parameters $\theta_1, \theta_t, \dots, \theta_k$, updating the belief p using the computed simulation errors, we explain here how the next simulation parameter θ_{k+1} is selected. Belief p implicitly defines another distribution P_{min} on the identity of the optimal model parameter θ^*,

$$P_{min}(\theta) \overset{def}{=} P\big(\theta = \arg\min_{\theta^i \in \Theta} E(\theta^i)\big) = \int_{E:\mathbb{R}^d \rightarrow \mathbb{R}} p(E) \Pi_{\theta^i \in \Theta - \{\theta\}} H\big(E(\theta^i) - E(\theta)\big) \mathrm{d}E,$$

where H is the Heaviside step function, i.e. $H\big(E(\theta^i) - E(\theta)\big) = 1$ if $E(\theta^i) \geq E(\theta)$ and $H\big(E(\theta^i) - E(\theta)\big) = 0$ else.

Unlike $p(E)$, the distribution of simulation error E modeled as a Gaussian Process, the distribution P_{min} does not have a closed-form expression. Therefore, we use *Monte Carlo* for estimating P_{min} from samples of $E(\theta^i)$ for each $\theta^i \in \Theta$. Specifically, we sample vectors containing the values that E takes, according to the learned Gaussian process, in each model parameter in Θ. $P_{min}(\theta^i)$ is estimated by counting the fraction of sampled vectors of the values of E where θ^i happens to have the lowest value.

We choose the model parameter θ that has the highest contribution to the current entropy of P_{min}, i.e. with the highest term $-P_{min}(\theta) \log \big(P_{min}(\theta)\big)$, as the next model parameter to evaluate in the simulation. We refer to this method as the *Greedy Entropy Search* method because it aims at decreasing the entropy of the belief P_{min}. This process is repeated until the entropy of P_{min} does not change much or until the simulation's time budget is consumed. After that, we use P_{min} as the new belief P_{t+1} on the model parameters. This new belief is used for planning an action μ_{t+1} which will move the object to a new pose x_{t+1}, and the same process is repeated all over again.

3 Experiment

3.1 Data Collection and Evaluation Metrics

We use a *Reflex SF* robotic hand mounted on a *Motoman SDA10F* arm to perform the real-world roll-outs. In this preliminary experiment, we learn the mass and the friction coefficient of a simple rigid box (an *Expo* eraser). Simtrack[8] is used to track the object and provide the initial and final poses of the object. Fifteen random poking actions were performed. Six were discarded due to inaccurate tracking caused by occlusions. Out of the remaining nine actions, we used six for training and the other three for testing. To measure the accuracy of the learned model, we compute the distance between the predicted pose of the object (3D position and 3D orientation) and the real observed pose.

3.2 Results

We compare the results of our Bayesian optimization method with random search in Figure 2. Random Search is performed by searching in the same parameter space as the Bayesian optimization. The time budget for both methods is 30 minutes. Both methods used the same three training samples and tested on the three test samples. During training, the objective function is the sum of three distances as defined in Sec. 3.1. We ran both methods ten times and reported the mean and stand deviation of training error. The result shows that Bayesian optimization achieved both lower error and faster convergence.

We also report the prediction error as a function of the number of training samples. We compare the prediction errors of models trained with one sample, three samples and all six samples in Figure 3.

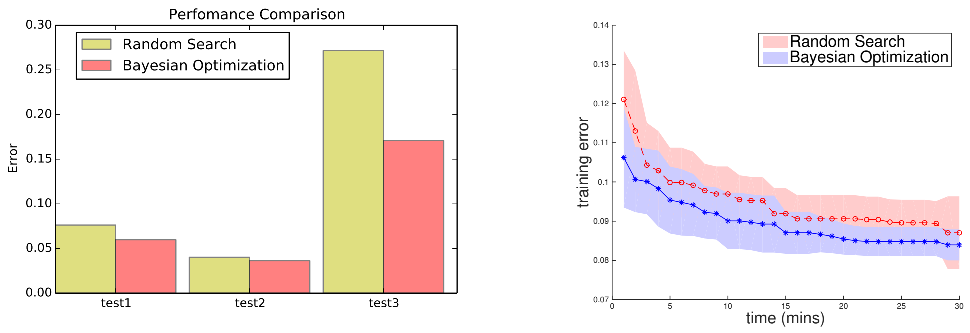


Figure 2: Comparison between Bayesian optimization method with random search. Bayesian optimization achieved both lower error and faster convergence.

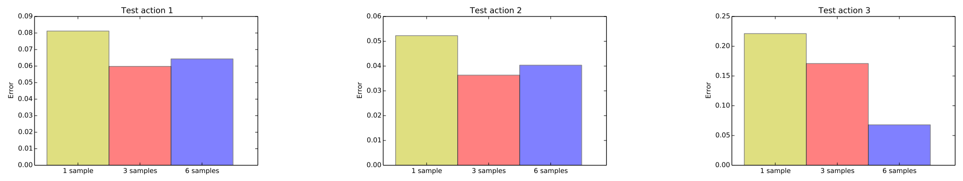


Figure 3: Pose prediction error as a function of the number of training samples

With more training samples, the error generally decreases. For test action 1 and 2, the error increased by a small margin when training with all six samples comparing to training with three. This could be because of the small size of the training set that one sample could lead to bias in the model. Increasing the size of the training data will be a focus for the next step of this work. Test results with the model trained with 6 samples can be found here: `http://bit.ly/2dRgut1`.

4 Future Work

The immediate next step in this work is learning model parameters with a larger number of examples. We are also considering experiments with various objects as well as utilizing existing large scale datasets[4, 10]. Additionally, we are investigating efficient ways for handling model parameters of non-homogenous objects. In this work, we considered only random exploratory actions, the learned predication model should eventually be used for control and action planning [4].

References

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

[2] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419*, 2016.

[3] A. Boularias, J. A. Bagnell, and A. Stentz. Efficient optimization for autonomous robotic manipulation of natural objects. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence,*, 2014.

[4] C. Finn and S. Levine. Deep visual foresight for planning robot motion. *arXiv preprint arXiv:1610.00696*.

[5] P. Hennig and C. J. Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[6] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, pages 430–438, 2016.

[7] W. Li, S. Azimi, A. Leonardis, and M. Fritz. To fall or not to fall: A visual approach to physical stability prediction. 2016.

[8] K. Pauwels and D. Kragic. Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[9] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[10] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez. More than a million ways to be pushed: A high-fidelity experimental data set of planar pushing. *arXiv preprint arXiv:1604.04038*, 2016.