Online Learning-Based Inertial Parameter Identification of Unknown Object for Model-Based Control of Wheeled Humanoids

Donghoon Baek[1], Bo Peng[2], Saurabh Gupta[3], and Joao Ramos[1,2]

***Abstract*—Identifying the dynamic properties of manipulated objects is essential for safe and accurate robot control. Most methods rely on low-noise force-torque sensors, long exciting signals, and solving nonlinear optimization problems, making the estimation process slow. In this work, we propose a fast, online learning-based inertial parameter estimation framework that enhances model-based control. We aim to quickly and accurately estimate the parameters of an unknown object using only the robot's proprioception through end-to-end learning, which is applicable for real-time system. To effectively capture features in robot proprioception solely affected by object dynamics and address the challenge of obtaining ground truth inertial parameters in the real world, we developed a high-fidelity simulation that uses more accurate robot dynamics through real-to-sim adaptation. Since our adaptation focuses solely on the robot, task-relevant data (e.g., holding an object) is not required from the real world, simplifying the data collection process. Moreover, we address both parametric and non-parametric modeling errors independently using *Robot System Identification* and *Gaussian Processes*. We validate our estimator to assess how quickly and accurately it can estimate physically feasible parameters of an manipulated object given a specific trajectory obtained from a wheeled humanoid robot. Our estimator achieves faster estimation speeds (around 0.1 seconds) while maintaining accuracy comparable to other methods. Additionally, our estimator further highlight its benefits in improving the performance of model based control by compensating object's dynamics and re initializing new equilibrium point of wheeled humanoid.**

***Index Terms*—Inertial Parameter Estimation, Real-to-Sim Adaptation, Representation Learning**

I. INTRODUCTION

Collaborative robots (e.g., Humanoid and manipulator) have become increasingly prevalent with great potential in various fields such as manufacturing, healthcare, and even disaster response [1], [2]. To facilitate seamless and safe human-robot collaboration, these robots need an accurate understanding of the physical properties of the objects they interact with. Understanding an object's inertial parameters (mass, center of mass, moment of inertia) enables robots to interact more robustly and adaptively. For instance, insufficient information may lead to excessive or insufficient force application, causing slipping or damage during manipulation tasks.

Identifying the inertial parameters of an unknown object with a humanoid robot is challenging due to several inherent factors: (1) noisy signals from force-torque sensors, (2) the need for long excitation trajectories to collect sufficient data, and (3) solving nonlinear constraint optimization for physically feasible parameters. These factors inherently slow down the estimation process, and the use of force-torque sensors and cameras is not always accessible.

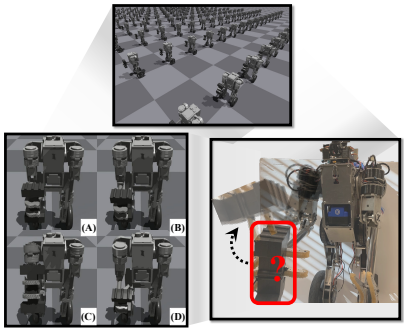


Fig. 1: **Conceptual overview of the proposed method.** Thousands of wheeled humanoid robots, SATYRR, in a simulator shake an object to identify its inertial parameters. Different object dynamics impact the robot's proprioception in varied ways. The effect of different object dynamics on the robot's proprioception closely mirrors real-world conditions due to effective real-to-sim adaptation. During training, the object's inertial parameters can be sampled either randomly or within specific shape boundaries.

In this paper, we propose a learning-based approach to quickly identify the inertial parameters of an unknown object, enhancing the performance of a model-based controller. Our method uses a data-driven regression model relying solely on proprioception errors influenced by object dynamics, not requiring long excitation signals and constrained nonlinear optimization. While using proprioception to identify parameters is not a new idea, *estimating the physically feasible full inertial parameters of unknown object in an end-to-end manner is a novel approach*. The main challenge is efficiently capturing the object's dynamic effects in the robot's proprioception while successfully transferring this information from simulation to the real world. For example, proprioception is influenced by the robot's own dynamics, the object's dynamics, and even the *reality gap*. To achieve this, we identify an accurate model of the robot to minimize its reality gap via real-to-sim adaptation, leaving only the effect of object dynamics on the robot's proprioceptive error. Unlike other approaches, we focus only on robot dynamics, eliminating the need to collect task-relevant data from the real world (e.g., the robot does not need to hold an object in the real world). Our adaptation addresses both parametric and non-parametric modeling errors using a combination of *Robot System Identification* (SysID) and *Gaussian Processes* (GPs). The data collection and training of the estimator are conducted offline and then transferred to the real world with zero-shot adaptation, ensuring fast inference

Manuscript received: August 4, 2024; Accepted October 04, 2024.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments.

This work is supported by the National Science Foundation via grant IIS-2024775.

The authors are with the [1] Department of Mechanical Science and Engineering and the [2] Department of Electrical and Computer Engineering and the [3] Department of Computer Science at the University of Illinois at Urbana-Champaign, USA.dbaek4@illinois.edu

times without additional iterations or tuning. We evaluate the estimation speed and accuracy of our estimator by benchmarking it against previous methods and also assessing the effects of the components in our adaptation method. Results show that our estimator can identify an object's inertial parameters within 0.1 seconds while achieving accuracy comparable to other baselines. Additionally, we show benefits of our method in improving the performance of model-based control by complementing object dynamics in manipulation tracking and delivering object locomotion tasks, achieving 36% and 65% performance improvements in each task, respectively.

Our summarized contributions are: (1) a fast inertial parameter estimation framework using an end-to-end data-driven model; (2) effective handling of both parametric and non-parametric modeling errors to identify accurate robot dynamics, reducing its *reality gap*; (3) demonstration on physical hardware, highlighting the benefits of accurate inertial parameters in enhancing model-based controller performance.

II. RELATED WORK

A. System Identification

Research on inertial parameter identification of rigid bodies has a rich history [3]. Since the dynamics model of a multi-body system is linear with respect to the inertial parameters, a linear least squares method has been widely used for this identification, whether through offline or adaptive means. However, this traditional approach has several limitations that must be further considered.

Without including constraints in the optimization process, not all combinations of parameters correspond to the physical system. Some studies have addressed this by incorporating constraints like custom manifold optimization [4], linear matrix inequalities (LMIs) [5], and Riemannian metrics [6]. These methods ensure physical consistency but are time-consuming (over 6 seconds) and require prior object knowledge (e.g., shape, CAD data). Our approach implicitly enforces dynamic consistency using regularization, achieving parameter estimation, including trajectory generation, in under 0.1 seconds.

Force-torque (FT) sensors are essential for determining an object's inertial parameters but are often hindered by weight, cost, and noisy outputs [7]. Alternatives like neural network-based force/torque estimation still require motor torque sensors [8]. A recent study [9] used encoder discrepancies and attention mechanisms to estimate mass and center of mass (COM) without FT sensors. Our approach builds on this by estimating the full set of inertial parameters without relying on torque/force sensors.

Learning-based inertial parameter estimation has been actively studied, often utilizing extensive visual datasets like images and videos [10], [11]. While accessing a visual dataset is not challenging, interaction with the object is crucial for estimating dynamics due to limited information like density. Our approach only relies on interacting with an object, without using cameras.

B. Sim-to-Real Transfer

With the recent advancements in reinforcement learning (RL), sim-to-real transfer has gained significant attention and is becoming increasingly crucial. Domain randomization, optimizing simulation parameters, and reducing parameter distribution [12], [13], [14], [15], [16], [17], [18] have been studied to reduce the *reality gap*. While these methods yield promising results, they require multiple iterations, making them unsuitable for real-time applications. Moreover, task-relevant data is often required in the training process, which is challenging to obtain, especially for the inertial parameters of an object. Similar work with ours can be found in [19] regarding using Gaussian Processes for robot system identification. In contrast, we employed these methods independently to address both parametric and non-parametric modeling errors. Some literature demonstrates the capability of estimating inertial parameters with deep neural networks [20]. While this method is fast and effective, it requires to use a pre-trained RL policy.

In the control perspective, approaches like domain randomization and privileged learning [21], [22] are designed for zero-shot policy transfer to the real world. While effective in reducing the *reality gap* for control, they have limited performance in matching state trajectories, which impacts parameter estimation accuracy [21].

III. BACKGROUND

A. Inertial Parameter Identification

The inertial parameters of i'th rigid-body are collected as $\boldsymbol{\phi}_i = [m_i, \mathbf{c}_i^\top, I_i^{xx}, I_i^{yy}, I_i^{zz}, I_i^{xy}, I_i^{yz}, I_i^{zx}]^\top \in \mathbb{R}^{10}$, where $m \in \mathbb{R}$ is the mass, $\mathbf{c} = [c_x, c_y, c_z]^\top \in \mathbb{R}^3$ is the the center of mass (COM) position, and $I^{(\cdot)}$ are the moments and the products of inertia. In the classical way, the inertial parameters of n_s-rigid-body system can be estimated by classical linear regression model $\mathbf{Y}(\mathbf{a}, \boldsymbol{\omega}, \dot{\boldsymbol{\omega}})$ using the rigid body dynamics,

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} m\mathbf{a} \\ [\boldsymbol{\omega}]\mathbf{I}\boldsymbol{\omega} + \mathbf{I}\dot{\boldsymbol{\omega}} \end{bmatrix} = \mathbf{Y}\phi. \tag{1}$$

where the symbolic $\mathbf{f} \in \mathbb{R}^3$ and $\boldsymbol{\tau} \in \mathbb{R}^3$ denote forces and torques, respectively. The term $m \in \mathbb{R}^+$ denotes the mass, $\mathbf{a} \in \mathbb{R}^3$ is the linear acceleration, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity, $\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the angular acceleration, and the bracket denotes $[\,\cdot\,]$ the skew-symmetric representation of a vector.

A simplistic estimation of inertial parameters with least-square regression may result in physically invalid parameters [5]. Hence, the regression could be enforced with physical consistency [4] as follows,

$$\min_{\mathbf{R}, \mathbf{J}, \mathbf{c}, m} \quad \sum_n \left\| \mathbf{Y}^{(n)} \pi(\mathbf{R}, \mathbf{J}, \mathbf{c}, m) - \tau^{(n)} \right\|^2 \tag{2a}$$

$$\text{s.t.} \quad \mathbf{R} \in SO(3), \tag{2b}$$

$$m > 0,\ J_i > 0,\ i = 1, 2, 3, \tag{2c}$$

$$J_1 + J_2 + J_3 \geq 2J_k, k = 1, 2, 3 \tag{2d}$$

to enforce the manifold constraint (2b), positive-semidefinitene of mass and inertia tensor (2c) and a triangular inequality(2d), where $J_{(\cdot)}$ denotes the eigenvalues of the inertia tensor of a rigid body ($\mathbf{I} = \mathbf{RJR^T}$). The solution to this problem is possible using nonlinear optimization on manifolds. In our case, we implicitly consider dynamic consistency by using the regularization (see Eq. 11).

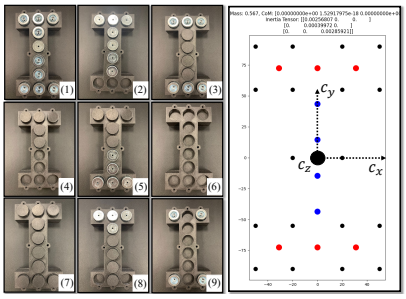


Fig. 2: **Customized Object To Get Ground-Truth Inertial Parameters.** Depending on the location of the weights, the object can be represented as a barbell(3), hammer(8), etc. Right side image shows a sample of an object(3) with ground truth inertial parameters. The CoM is defined based on a central fixed coordinate system.

B. Real-to-Sim Adaptation

In terms of dynamics, the *reality gap* δ is caused by a combination of inaccurate parameters in the parametric model and non-parametric uncertainties (e.g., backlash, hysteresis, etc). The *reality gap* δ regarding the nonlinear dynamics of a multi-body system can be decomposed into a parametric modeling error $\delta\hat{f}$ and non-parametric modeling error $\delta\hat{g}$ as follows:

$$\delta_t = \underbrace{\delta\hat{f}(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t}; \boldsymbol{\zeta})}_{\text{Parametric Error}} + \underbrace{\delta\hat{g}(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t})}_{\text{Non-Parametric Error}} \tag{3}$$

where the system parameters $\boldsymbol{\zeta}$, the estimated states $\hat{\mathbf{x}}$, the control effort $\hat{\mathbf{u}}$, and time t. To compensate for the error caused by $\delta\hat{f}$ and $\delta\hat{g}$, we leverage the SysID and GPs, respectively.

In this work, we handle the errors $\delta\hat{f}$ and $\delta\hat{g}$ separately, ensuring that each error is corrected in its own step rather than optimizing both simultaneously. Note that this approach focuses on reducing the *reality gap* for a robot itself without considering object dynamics, eliminating the need for real object datasets during training, but still leaves a *reality gap* in object dynamics.

IV. METHODOLOGY

Our framework consists of two distinct steps: 1) Real-to-Sim Adaptation aimed at enhancing the fidelity of the simulation via *Robot System Identification* (SysID) and *Gaussian Processes* (GPs); and 2) Learning the inertial parameters for an unknown object via time-series data-driven regression model. As a preliminary step for developing the end-to-end inertial parameter estimation using a high-fidelity simulation, we assume that the object can be perfectly held by the robotic hand, and a single trajectory is leveraged [19]. The overview of the proposed method is described in Algorithm 1.

A. Target System Description and Unknown Object Design

In this work, we utilize a four degrees-of-freedom (DoF) manipulator of a wheeled humanoid, SATYRR [1], to hold and shake an object. To focus on manipulation part, we fix its torso to minimize the effect of the dynamics of whole-body SATYRR. Inspired by the previous work [2], our object is designed to easily and precisely calculate the ground truth inertial parameter of various objects. This object consists of a combination of three cuboids, ten-cylinder holes, and steel weights. These weights can be placed in ten different locations as depicted in Fig.2. The ground truth of inertial parameters can be calculated based on the moment of inertia, the density of each weight, and the size of each cuboid.

B. Real-to-Sim Adaptation

1) Offline Robot System Identification: The goal of SysID is to minimize the discrepancy between a source data $\mathbf{X_S}$ and a target data $\mathbf{X_T}$ by searching for more realistic robot's system parameters $\boldsymbol{\zeta}$ in a simulation. The cost function can be some distance function $d(\cdot,\cdot)$ over a dataset of m samples where T is the length of each sample (Eq. (4)). We chose the mean square error (MSE) as a d function that is commonly used in SysID [5].

$$\mathcal{L} = \arg\min_{\boldsymbol{\zeta}} \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T} d(\mathbf{X_S^t}, \mathbf{X_T^t}). \tag{4}$$

The optimal parameter $\boldsymbol{\zeta}^*$ can be chosen by solving the Eq. 4 and we utilized the particle swarm optimization (PSO) algorithm due to its global search ability and fast convergence speed. In this work, data $\mathbf{X} \in \mathbb{R}^8$ contains position q and velocity $\dot{q}$ in four joints of the manipulator in SATYRR. The parameters $\boldsymbol{\zeta} \in \mathbb{R}^8$ to be optimized contain joint damping

Algorithm 1 Procedure of Learning Inertial Parameter Estimation of unknown Object via Real-to-Sim Adaptation

Phase 1 *Robot System Identification*
Input: Target data $\mathcal{D}_\mathcal{T}$ (Free case - not holding an object)
Initialize: $\boldsymbol{\zeta} \in \mathbb{R}^8$, $\mathcal{L} \in \mathbb{R}$
while until ($\mathcal{L}$ converges) **do**
  $\mathbf{X_T} = \{q^{1:4}, \dot{q}^{1:4}\} \leftarrow \mathcal{D}_\mathcal{T}$
  $\mathbf{X_S} = \{\hat{q}^{1:4}, \hat{\dot{q}}^{1:4}\} \leftarrow \hat{f}(\cdot;\boldsymbol{\zeta})_{sim}$ ▷ IssacGym
  Calculate the Loss function $\mathcal{L}$ ▷ Eq. (4)
  Update parameter $\boldsymbol{\zeta}$ via *particle swarm optimization*
  Return: $\boldsymbol{\zeta}^*$
end while

Phase 2 *Gaussian Processes*
Input: *Gaussian Processes* $\mathcal{GP}$, Simulation with SysID applied $\hat{f}(\cdot;\boldsymbol{\zeta}^*)_{sim}$, and $\mathcal{D}_\mathcal{T}$
do
  $\mathbf{X_T} = \{q^{1:4}, \dot{q}^{1:4}\} \leftarrow \mathcal{D}_\mathcal{T}$
  $\mathbf{X_S} = \{\hat{q}^{1:4}, \hat{\dot{q}}^{1:4}\} \leftarrow \hat{f}(\cdot;\boldsymbol{\zeta}^*)_{sim}$
  Calculate non-parametric modeling error $\hat{g}(\hat{\mathbf{x}}, \hat{\mathbf{u}}, t)$
  $(\hat{g}(\hat{\mathbf{x}}, \hat{\mathbf{u}}, t) = \mathbf{X_T} - \mathbf{X_S})$
  Optimize a GP regression model using GPy. ▷ Eq. (5)
  Save a GP model $\mathcal{GP}(\mu, k)$
  Apply the $\mathcal{GP}(\mu, k)$ to the simulation $\hat{f}(\cdot;\boldsymbol{\zeta}^*)_{sim}$
  Return: $\hat{f}(\cdot;\boldsymbol{\zeta}^*, \mathcal{GP})_{sim}$
end

Phase 3 Learning Inertial Parameter of an Object
Input: $\mathcal{D}_\mathcal{S} \leftarrow \hat{f}(\cdot;\boldsymbol{\zeta}^*, \mathcal{GP})_{sim}, \mathcal{L}_{\mathcal{NN}}$
while until ($\mathcal{L}_{\mathcal{NN}}$ converges) **do**
    Training a time-series data-driven model IV-C3
    to minimize the loss $\mathcal{L}_{\mathcal{NN}}$ ▷ Eq. (11)
end while

$\mathbf{d_j} \in \mathbb{R}^4$ and link mass $\mathbf{m_l} \in \mathbb{R}^4$. These parameters are chosen based on the major components that impact the dynamics. The center of mass and inertia tensor are determined using the CAD file, after setting the material and the coordinate center. To create a realistic simulation environment, we set the PD controller gains ($\mathbf{K_p} \in \mathbb{R}^4, \mathbf{K_d} \in \mathbb{R}^4$) based on the real-world gains used for SATYRR, with a control frequency of 400Hz for both simulation and real-world scenarios.

2) Non-Parametric Dynamics Modeling via Gaussian Processes: To model the reality gap caused by non-parametric modeling error $\delta\hat{g}(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t})$, such as nonlinear friction and backlash, we leverage Gaussian Processes (GPs). GPs offer the benefit of sample efficiency and the potential to construct a wide range of functions without assuming a specific functional form [23]. Given the different scales between joint position and velocity, we employed separate GP regression models to accurately depict the residual errors for joint position and velocity, respectively. (e.g., output of GPs $y = \mathbf{X_T} - \mathbf{X_S}$ and $\mathbf{X_S}$ is acquired from a simulation $\hat{f}(\cdot; \boldsymbol{\zeta}^*)_{sim}$ which utilizes the optimal parameters $\boldsymbol{\zeta}^*$ from SysID). The non-parametric dynamics model $\hat{g}$ can be formulated with GPs as follows:

$$\hat{g}(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t}) = \mathcal{GP}(\mu(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t}), k(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t}, \hat{\mathbf{x}}'_{\boldsymbol{t}'}, \hat{\mathbf{u}}'_{\boldsymbol{t}'})). \tag{5}$$

here, the individual terms represent the state $\hat{\mathbf{x}}$, control effort $\hat{\mathbf{u}}$, and time t. The GPs $\mathcal{GP}(\mu, k)$ is expressed with a mean function μ and covariance function k where the mean $\mu(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t})$ represents the expected value of the process at each point in the input space and the covariance function $k(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{u}}_\mathbf{t}, \hat{\mathbf{x}}'_{\boldsymbol{t}'}, \hat{\mathbf{u}}'_{\boldsymbol{t}'})$ describes the correlation between outputs for two distinct sets of inputs in the function $\hat{g}$. We constructed and optimized the GP regression model using the GPy library. Radial Basis Function (RBF) kernel (both variance and length scale set to 1 for position and 50 for velocity) is utilized. The RBF kernel parameters were manually selected until the *reality gap* was sufficiently reduced (see Fig. 5).

C. Learning Inertial Parameter of Unknown Object

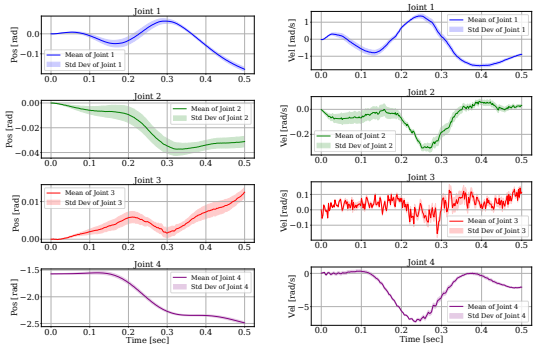
1) Dynamic Trajectory Planning and Manipulation Control: Periodic excitation trajectories, often based on Fourier series and trigonometric functions, are commonly used for dynamic model identification [5], [2], [24]. While effective, they can be time-consuming (e.g., 35 seconds in [2]). Inspired by how humans shake objects to identify them (with decreasing amplitude and frequency over time), we designed the dynamic trajectory as follows.

$$f(t) = h_{\text{freq}} - (h_{\text{freq}} - l_{\text{freq}})t/T, \qquad \phi(T) = \Delta t \sum_{\tau=0}^{T/\Delta t} f(\tau) \tag{6}$$

$$w(t) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi t}{T}\right)\right) \tag{7}$$

$$X(t) = \alpha \sin(2\pi\phi(T))w(t) \tag{8}$$

where the frequency $f(t)$ starts at h_{freq} and linearly decreases to l_{freq}. The signal phase, ϕ, is computed as the cumulative sum of the frequency, scaled by the time step. The sinusoidal trajectory $X(t)$ has an amplitude scaled by a factor α and is



(a) Joint-level Position (b) Joint-level Velocity

Fig. 3: **Input Data Distribution in Training Dataset**. The graphs shows the mean and standard deviation of joint trajectories in training dataset. This represents how different object's dynamics properties affect the trajectories of joint position and velocity for each joint.

modulated by a *Hann window*, $w(t)$, for smooth transitions. The trajectory $X(t)$ is applied to the end-effector in x and y axis ($T = 0.5$, x-axis: $h_{freq} = 5, l_{freq} = 1, \alpha = -1$, y-axis: $h_{freq} = 3, l_{freq} = 1, \alpha = 5$). The numerical inverse kinematics using a pseudo-inverse Jacobian of the 4-DoF manipulator is employed to control the manipulator of SATYYR, which is defined as

$$\begin{aligned} J^+ &= (J^T J + \lambda^2 I)^{-1} J^T, \\ \theta_{des} &= \theta + \lambda J^+ e. \end{aligned} \tag{9}$$

where J^+ represents a pseudo-inverse Jacobian leveraging a *Damped Least Squares method*, aiding to improve stability near a singular configuration and dealing with noisy data or slight modeling errors by utilizing regularization. The symbol λ is damping, α is a hyperparameter, and $e = x_{des} - x$ is task-space position error.

2) Database Construction: To build a learning-based object dynamics estimator, we constructed the M number of source dataset $\mathcal{D}_S = (\mathbf{X}_S^i, \mathbf{y}_S^i)_{i=1}^M (M = 5000)$ using the IsaacGym simulator [25], with 5,000 environments running simultaneously. To directly apply the pre-trained estimation model to the SATYRR without further manual tuning, we employed a more realistic simulator to acquire the dataset $\mathcal{D}_S$. All agents in the simulator track the desired trajectory (Eq. 8) while holding different objects. The estimator takes as input the concatenated vector $\mathbf{X_S}$ and outputs estimated inertial parameter $\mathbf{y_S}$ as:

$$\begin{aligned} \mathbf{X_S} &= [q^1_{t:T}, q^2_{t:T}, q^3_{t:T}, q^4_{t:T}, \dot{q}^1_{t:T}, \dot{q}^2_{t:T}, \dot{q}^3_{t:T}, \dot{q}^4_{t:T}]^\top \\ \mathbf{y_S} &= [m, c_x, c_y, c_z, I_{xx}, I_{yy}, I_{zz}]^\top \end{aligned} \tag{10}$$

where $q^i_{t:T}$ and $\dot{q}^i_{t:T}$ are the history of joint position and velocity of each joint, respectively. The term $\mathbf{y_S}$ denotes the vector of inertial parameters. Each parameter component is either 1) randomly generated with a uniform distribution, or 2) determined by adjusting the weight location, length scale (width and height of each cuboid), and density. After random generation, we verified their mass and ensured the inertia tensor satisfies the triangular inequality for physical feasibility. We simplify the problem by focusing on estimating the diagonal terms in the inertia matrix considering a more tractable analysis while retaining the essential physics of the problem. To collect real-world data for validation, the wheeled

humanoid robot SATYRR [1] tracked a pre-recorded shaking motion generated in simulation (see Eq. (9)). A total of ten different objects were involved, along with a free case where no object was held. For each object case, the trajectory was applied to the SATYRR five times (i.e., a total of 45 real-world dataset were obtained for the validation in the experiment).

3) Training Time-Series Data-Driven Model with Dynamic Consistency Regularization: To learn the inertial parameter ϕ, we trained a time-series data-driven regression model (e.g., 1D-CNN) using 4000 samples of the training set in dataset $\mathcal{D}_S$ (500 for validation set and 500 for test set). Dynamic consistency is implicitly taken into account by applying the regularization term in the loss function to estimate more physically reliable parameters. The total loss function considering dynamic consistency is defined as

$$\begin{aligned}
\mathcal{L}_{\mathcal{NN}} &= w_1 \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y_S^i} - \hat{\mathbf{y}}_\mathbf{S}^\mathbf{i})^2 + w_2\mathcal{L}_{\text{tri}} + w_3\mathcal{L}_{\text{pos}}\hat{\mathbf{y}}_\mathbf{S} \\
\mathcal{L}_{\text{tri}}(I) &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{3}\text{ReLU}\left(I_j^i - \sum_{k=1}^{3} I_k^i\right) \\
\mathcal{L}_{\text{pos}}\hat{y} &= \frac{1}{n}\sum_{i=1}^{n}\left|\min\left(\hat{\mathbf{y}}_\mathbf{S}^\mathbf{i}(k), 0\right)\right|,\ k = [0, 4, 5, 6]
\end{aligned} \tag{11}$$

where the total loss $\mathcal{L}_{\mathcal{NN}}$ combines mean square error, triangular inequality loss $\mathcal{L}_{\text{tri}}(I^{(\cdot)})$, and negative output penalization loss $\mathcal{L}_{\text{pos}}\hat{\mathbf{y}}_\mathbf{S}$. The term w_1, w_2, and w_3 are the weight of each term and are decided by manual tuning considering the importance of each parameter. We have empirically observed that neural network is capable of learning inertial parameters without accounting for the varying scales of these parameters. Our 1D CNN architecture processes sequential data with four 64-filter convolutional layers (kernel size 3, stride 1, padding 1), ReLU activations, and a MaxPooling layer (kernel size 2). It concludes with a dense network transitioning from 512 to 128 neurons, leading to the specified output size.

V. EXPERIMENT

The proposed learning-based inertial parameter identification method is verified with a 4-DoF manipulator of SATYRR [1] in both a simulation and the real world. The experimental setup is illustrated in Fig. 1 and Fig. 2. Pre-defined trajectory, SATYRR robot, HMI, and ground-truth of inertial parameters are utilized. We conducted three major experiments in total. Details are described below.

A. Real-to-Sim Adaptation

The main purpose is to validate our sim-to-real adaptation by answering two questions: (1) How effectively can SysID and GPs compensate for parametric and non-parametric modeling errors to reduce the reality gap? and (2) Does this method generalize to test environments where object dynamics are taken into account? We evaluate the real-to-sim adaptation ability by comparing six separate baselines: (a) **PureSim**: simulation using default physics parameters and PD controller with almost perfect tracking performance (b) **Sim+ActNet**: using Actuator Network [22] instead of PD controller. We collected a dataset from SAYTRR and trained the Actuator Network parameterized with θ defined as

$$\tau_j = h_\theta(q_j^*, q_j, \dot{q}_j^*, \dot{q}_j, K_p, K_d), \qquad j = 1, 2, 3, 4 \tag{12}$$

where desired joint position and velocity q^* and $\dot{q}^*$, actual joint position and velocity q and $\dot{q}$, and controller gain K_p and K_d. (c) **Sim+GPs**: applying the GPs into PureSim (d) **Sim+SysID**: applying the SysID into PureSim (e) **Sim+SysID+GPs**: Applying the GPs to (d) (f) **ActNet+Gps**: applying the GPs to (b). The mean sqaure error (MSE, Eq. 4) is used as an evaluation index. Note that the GP model is trained exclusively on data without objects, yet it's tested under conditions where the robot interacts with objects. This tests the adaptability of the optimized SysID and GPs to unseen dynamics, evaluating their capability to accurately estimate object inertial parameters in new scenarios.

B. Inertial Parameter Estimation

This experiment addresses the following questions: (1) How fast and accurate is our framework compared to previous methods for estimating the inertial parameters of unknown objects? Which neural network models are most suitable? (2) How does our method perform in real-world scenarios, and does our adaptation method improve estimation performance? (3) How does our method compare to previous works, and can it be applied to a more generalizable object dataset? We evaluated the performance of our framework against the following baselines: (1) Ordinary Least Squares (OLS) [3] and (2) Weighted Least Squares (WLS) [24]. Since these traditional methods require a force/torque sensor, we used 45 samples from the simulation test dataset to directly assess force and torque values. For a fair comparison, we used the open-source MATLAB code provided by the authors [3], [24]. We also adopted TuneNet [16], DROPO [13] and OSI [20] as learning-based baseline methods. The mean absolute error (MAE=$|\hat{y}_k - y_k|$) and the normalized MAE (NMAE= $\frac{1}{n}\sum_{k=1}^{n}\frac{|\hat{y}_k - y_k|}{y_{\text{scale}}}$) are leveraged as evaluation indexes which are commonly chosen in this field [9], [7].

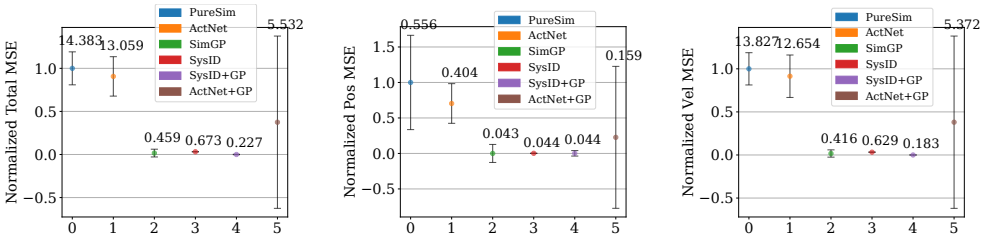
C. Control Task Experiments

How beneficial are the identified inertial parameters for improving model-based controller performance, even with imperfect estimation accuracy? To verify this, we conducted real-world manipulation and locomotion tasks using a SATYRR robot holding a 1.5 kg drill (see Fig. 7). In the manipulation task, we assessed the robot's trajectory tracking with the object, which is essential for warehouse tasks. For locomotion, we evaluated the robot's proficiency in transporting the object.

VI. RESULTS AND DISCUSSION

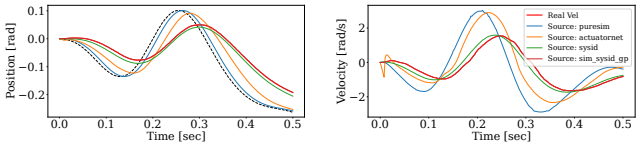
A. Real-to-Sim Adaptation

As shown in the Fig 5, the SysID and GPs contribute to reduce the *reality gap* in both position and velocity trajectories. In the case of GPs, it learns not only non-parametric error but also noise in real data.



(a) Total: Position+Velocity (b) Position (c) Velocity

Fig. 4: **Results of Real2Sim Adaptation**. The graphs depict the mean and standard deviation of the normalized Mean Squared Error (MSE) between trajectories derived from simulation and the real world, employing min-max normalization for straightforward comparison. The 45 target objects, which are not considered in the optimization process, are utilized for evaluation. The numbers on the graph represent the MSE. Based on the normalized total MSE outcomes, **Sim+SysID+GP** exhibits the smallest *reality gap*. The MSE for **Sim+SysID** is also notably low, showing only a minor performance gap compared to **Sim+SysID+GP**. This is because the error of parametric model has a large portion in causing the *reality gap*. The error from non-parametric modeling can become substantially larger in scenarios involving contact.



(a) Joint 1 Position (b) Joint 1 Velocity

Fig. 5: **Trajectory Comparison Results Obtained From Real World and Simulation**. The trajectory produced by **Puresim** is closely located with the desired trajectory (black). First, **SysID** is applied to **PureSim**, followed by **GP** to **SysID** to handle the errors in parametric and non-parametric modeling, respectively. Each step brings the simulation trajectory closer to the target trajectory (red). In the case of **Sim+SysID+GP**, all joint trajectories in position and velocity are almost perfectly matched with the actual trajectories.

Although we successfully reduced the *reality gap* in the robot system, residual dynamics errors caused by the object's dynamics still remain. The results illustrated in Figure 4 shows that our method, developed without object dynamics information, performs effectively even in a situation where new object is applied. Implementing the SysID on a **PureSim** environment significantly diminishes the *reality gap* for all examined object cases. Note that none of objects are utilized in optimizing the SysID and GPs. Overall, **Sim+SysID+GP** exhibits the smallest *reality gap* in comparison to other approaches. It has been observed that incorporating SysID contributes to a more generalizable adaptation from real to simulated environments. This is evidenced by the superior performance of **Sim+SysID+GP** over the use of **Sim+GP**, even though **Sim+GP** also perfectly reduce the reality gap within the training set. This supports that the parametric model can be more generalizable than non-parametric model using the GPs model. While the Actuator Network demonstrated potential in narrowing the *reality gap* [22], it fell short in aligning the trajectories between the simulated and real environments. This discrepancy suggests that while the Actuator Network can help RL algorithms learn more realistic actions based on more accurate state information, it does not guarantee a complete convergence of trajectories across the two domains. The similar result can be observed in [21].

While the **Sim+SysID+GP** approach has shown effectiveness in closing the reality gap across different objects, it still requires enhancements in several aspects. The real-to-sim adaptation relies on a single trajectory, making its effectiveness highly dependent on the selected trajectory. This constraint is particularly pronounced due to the reliance on GPs [19] and the trajectory used for SysID. It is well-known that the performance of data-driven models is significantly influenced by the training data distribution. Exploring and combining physics-based and data-driven models could be an interesting direction for future research.

B. Inertial Parameter Estimation

1) Benchmark Results with Simulation Dataset: We validated the proposed inertial parameter estimation framework against conventional methods in terms of accuracy, speed, and dynamic consistency (see Fig. 6). The simulation dataset is employed to obtain crucial measurements such as acceleration, torque, and force at the object level, which are fundamental elements in conventional methods. The proposed estimator using a 1D-CNN achieved the highest accuracy in estimating the inertial parameter of unknown objects with SATYRR. As shown in Table III, the total estimation time for our method is around 0.1 second including generating a trajectory (e.g., arm shaking motion). On the other hand, we observed that the existing methods such as OLS ans WLS took more time as they depend on a long persistently exciting signal, highlighting our method's potential for real-time object characteristic identification. The 1D-CNN surpassed LSTM and TCN in estimation performance, efficiently capturing local dependencies via Convolutional filters. This is because the distinct behaviors associated with dynamics are likely concentrated within specific segments of the entire trajectory. This localization makes the 1D-CNN particularly adept at identifying these critical features for enhanced performance.

2) Estimation Results with Real World Dataset: In the real-world estimation of inertial parameters (see Table I), both **Sim+SysID** and **Sim+SysID+GP** models exhibit superior estimation performance, underscoring the effectiveness of real-to-sim adaptation. While the results are not as high as those achieved in simulations (shown in Fig. 6), they surpass traditional methods in estimating the CoM and inertia. In contrast, the **PureSim**, **Sim+GP**, and **Sim+ActNet** models demonstrate overfitting to simulation data, which is influenced by their

TABLE I: **Results of Inertial Parameter Estimation of Unknown Objects In the Real-World.** The numbers (e.g., A (B)) represent the mean (A) and standard deviation (B) of the estimation performance of our estimator (1D-CNN case) for nine different objects, as shown in Fig. 2 of our manuscript. We excluded the results of the **PureSim**, **Sim+GP**, and **Sim+ActNet** models due to overfitting. The physically feasible value indicates the rate of non-feasible outcomes out of the total number of trials (zero means all outputs are physically feasible).

	Mean Absolute Error (MAE)							Normalized Mean Absolute Error (NMAE)							
Inertia Parameters	Mass (kg)	CoM x (m)	CoM y (m)	CoM z (m)	Ixx (kgm^2)	Iyy (kgm^2)	Izz (kgm^2)	Mass (kg)	CoM x (m)	CoM y (m)	CoM z (m)	Ixx (kgm^2)	Iyy (kgm^2)	Izz (kgm^2)	Physically Feasibility
Sim +ActNet+GP	0.286 (0.18)	0.004 (0.001)	0.039 (0.015)	0.023 (0.001)	0.002 (0.003)	0.004 (0.003)	0.002 (0.001)	0.55 (0.24)	4.60 (1.96)	23.6 (20.44)	0.92 (0.06)	1.20 (1.38)	7.25 (5.00)	1.73 (1.24)	9/45
Sim + SysID	0.216 (0.13)	0.002 (0.001)	0.011 (0.008)	0.001 (0.0001)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)	0.41 (0.21)	1.63 (1.02)	4.68 (4.90)	0.05 (0.005)	0.66 (0.6)	2.2 (1.96)	0.54 (0.5)	8/45
Sim+SysID +GP	0.293 (0.35)	0.001 (0.0004)	0.010 (0.008)	0.0009 (0.001)	0.001 (0.002)	0.0005 (0.001)	0.001 (0.002)	0.63 (0.53)	1.09 (0.44)	3.66 (3.51)	0.039 (0.04)	0.86 (0.9)	0.78 (1.16)	0.94 (0.92)	1/45

TABLE II: **Benchmark Results of Parameter Estimation In the Real-World.** The numbers represent the same meaning in Table I. In this experiment, we trained our method with a new dataset, where each inertial parameter was randomly sampled with a uniform distribution, independent of the object's shape. The OSI is trained with our SysID+GPs dataset. We employed TuneNet (Obs) with three iterations.

	Mean Absolute Error (MAE)							Normalized Mean Absolute Error (NMAE)								
Inertia Parameters	Mass (kg)	CoM x (m)	CoM y (m)	CoM z (m)	Ixx (kgm^2)	Iyy (kgm^2)	Izz (kgm^2)	Mass (kg)	CoM x (m)	CoM y (m)	CoM z (m)	Ixx (kgm^2)	Iyy (kgm^2)	Izz (kgm^2)	Physical Feasible	Inference Time (s)
Ours	0.176 (0.16)	0.013 (0.003)	0.010 (0.009)	0.026 (0.0007)	0.002 (0.0009)	0.005 (0.0015)	0.002 (0.001)	0.357 (0.274)	13.41 (3.803)	2.749 (4.904)	1.045 (0.03)	1.352 (0.999)	10.37 (4.997)	1.307 (1.036)	3/45	0.102
OSI	0.243 (0.172)	0.049 (0.011)	0.032 (0.013)	0.053 (0.019)	0.052 (0.007)	0.041 (0.005)	0.075 (0.0067)	0.478 (0.277)	49.03 (11.89)	18.55 (15.6)	2.140 (0.785)	29.41 (12.27)	80.48 (32.68)	44.76 (19.81)	0/45	0.100
TuneNet (Obs) [16]	0.3 (0.2)	0.004 (0.0001)	0.015 (0.007)	0.025 (0.0004)	0.0008 (0.0004)	0.0007 (0.0006)	0.0008 (0.0001)	0.744 (0.490)	4.828 (0.15)	9.425 (7.704)	1.020 (0.016)	0.368 (0.196)	1.474 (0.859)	0.355 (0.16)	0/45	7-8
DROPO [13]	0.34 (0.21)	0.009 (0.001)	0.014 (0.009)	0.026 (0.009)	0.0025 (0.0008)	0.0025 (0.0001)	0.0024 (0.0009)	0.965 (0.889)	9.812 (1.031)	6.056 (4.428)	1.061 (0.380)	1.444 (0.969)	4.697 (4.39)	1.438 (1.014)	11/45	10 <

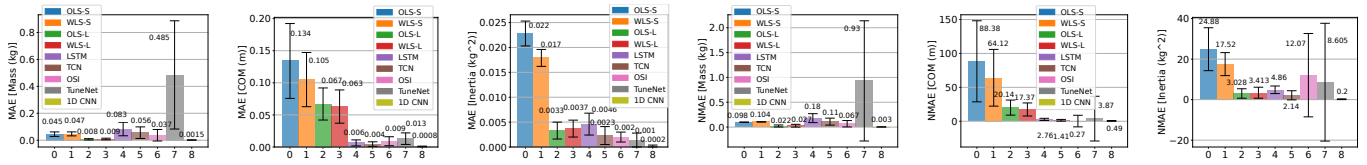


Fig. 6: **Inertial Parameter Estimation Benchmark in Simulation**. Overall, 1D-CNN showed most accurate estimation performance compared to other baselines. The TuneNet exhibited poor performance in estimating mass, while demonstrating comparable performance in estimating the center of mass (CoM) and inertia. Learning-based methods generally outperformed conventional least-squares methods, highlighting their potential to achieve fast and accurate parameter estimation.

TABLE III: **Runtime Benchmark in Simulation.** All learning-based methods demonstrated rapid estimation speeds, with runtimes around 0.1 second. Traditional methods take much time due to the persistently excitation signal.

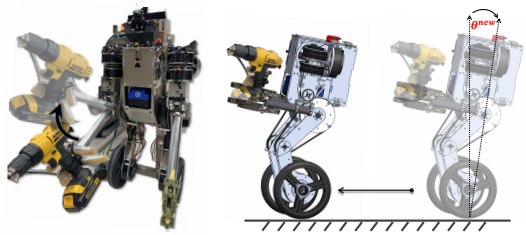
Method	OLS(S)	WLS(S)	OLS(L)	WLS(L)	LSTM	TCN	1D-CNN
Time (s)	0.525	0.56	10.09	10.4	0.102	0.105	0.102

limited sim-to-real adaptation ability. (see Fig. 4). We observed that using a GP slightly improves estimation performance compared to **Sim+SysID**, likely because GPs are not trained with specific object information, limiting their generalization. Additionally, the noise inherent in the signals captured by the GPs could impede the extraction of crucial features needed for accurate estimation. Despite applying soft regularization to ensure physically feasible solutions, 44 out of 45 solutions of **Sim+SysID+GP** meet the criteria for physical consistency, whereas most solutions from existing methods using non-constrained optimization did not satisfy physical consistency.

3) Benchmark Results Using More Generalizable Dataset: We reported benchmark performance comparison results in Table III. Our proposed has no noticeable decrease in performance even when trained on a more generalizable dataset with uniformly and independently selected parameters. Training with a uniform distribution for each parameter independently helps estimate mass due to its wide coverage, but is less effective for estimating the center of mass (CoM) and inertia, which are more influenced by the object's shape. Our method achieves comparable accuracy to baselines like TuneNet and DROPO, which require multiple iterations for parameter estimation, but ours operates significantly faster. While additional iterations can enhance accuracy, they also increase inference time, rendering them unsuitable for real-time applications. Since we handle the adaptation entirely offline and separate it from the actual evaluation step, it provides advantages in both speed and accuracy. Specifically, we reduce the *reality gap* of the robot system offline and then train a data-driven estimator on this adapted model. In contrast, other methods iteratively tune simulation parameters to reduce the *reality gap*.

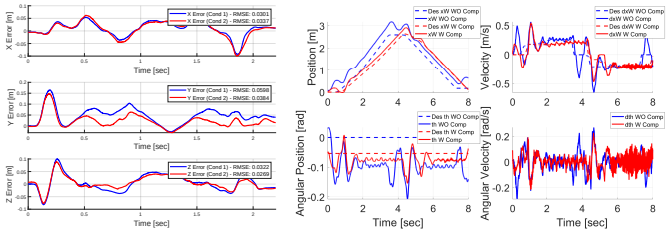
C. Control Task Experiments

1) Manipulation Task: After estimating an object's inertial parameters, we use these parameters for gravity compensation, effectively demonstrating the benefits of explicit parameter utilization. As illustrated in Fig. 8, there is a notable 36% performance enhancement along the y-axis, where gravity is exerted. Transitioning to a controller based on impedance control or inverse dynamics, which rely more heavily on model accuracy, promises even greater performance improvements in handling dynamic environments and interactions.



(a) Manipulation task with holding a drill (b) Locomotion task with holding a drill

Fig. 7: **Control Task Experiments**. The recorded motion obtained from a Human machine interface (HMI) is used as a desired trajectory in a manipulation task. The forward and backward driving tests were conducted using HMI (refer to the attached video).



(a) Manipulation Task Result (b) Locomotion Task Result

Fig. 8: **Results of Task Experiments**. Integrating the estimated inertial parameters of an object with a baseline controller significantly improves tracking performance for both manipulation and locomotion tasks. The estimated parameters are as follows: $\mathbf{y_S} = [1.1, -0.008, 0.1, 0.025, 0.005, 0.0007, 0.005]^\top$.

2) Locomotion Task: We integrated the estimated parameters into the baseline controller by recalculating the SATYRR robot's equilibrium point, considering the combined CoM and mass. This updated equilibrium point was used as the desired pitch angle in an LQR controller, resulting in a 65% improvement in position tracking (Fig. 8). Even greater performance gains can be expected with model-predictive or whole-body controllers, which rely more on model accuracy.

VII. CONCLUSION

In this paper, we propose a fast online learning-based framework to identify the inertial parameters of unknown objects, enhancing the accuracy of model-based controllers and making them more suitable for real-time applications. We introduce a real-to-sim adaptation that combines *Robot System Identification* and *Gaussian Processes* to reduce the *reality gap* caused by parametric and non-parametric modeling error, respectively. The adaptation method maintained effective performance even with new objects attached, and our estimation framework is significantly faster than other methodologies. Exploring the impact of accurate system parameter estimation on RL policy would be an interesting future research direction.

REFERENCES

[1] A. Purushottam, Y. Jung, K. Murphy, D. Baek, and J. Ramos, "Hands-free Telelocomotion of a Wheeled Humanoid," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8313–8320.

[2] P. Nadeau, M. Giamou, and J. Kelly, "Fast Object Inertial Parameter Identification for Collaborative Robots," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3560–3566.

[3] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 101–119, 1986.

[4] S. Traversaro, S. Brossette, A. Escande, and F. Nori, "Identification of fully physical consistent inertial parameters using optimization on manifolds," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5446–5451.

[5] P. M. Wensing, S. Kim, and J.-J. E. Slotine, "Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 60–67, 2017.

[6] T. Lee and F. C. Park, "A geometric algorithm for robust multibody inertial parameter identification," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2455–2462, 2018.

[7] B. Sundaralingam and T. Hermans, "In-hand object-dynamics inference using tactile fingertips," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1115–1126, 2021.

[8] N. Yilmaz, J. Y. Wu, P. Kazanzides, and U. Tumerdem, "Neural network based inverse dynamics identification and external force estimation on the da Vinci Research Kit," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1387–1393.

[9] Z. Lao, Y. Han, Y. Ma, and G. S. Chirikjian, "A Learning-Based Approach for Estimating Inertial Properties of Unknown Objects from Encoder Discrepancies," *IEEE Robotics and Automation Letters*, 2023.

[10] T. Standley, O. Sener, D. Chen, and S. Savarese, "image2mass: Estimating the mass of an object from its image," in *Conference on Robot Learning*. PMLR, 2017, pp. 324–333.

[11] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," *Advances in neural information processing systems*, vol. 28, 2015.

[12] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.

[13] G. Tiboni *et al.*, "DROPO: Sim-to-real transfer with offline domain randomization," *Robotics and Autonomous Systems*, vol. 166, p. 104432, 2023.

[14] Y. Du *et al.*, "Auto-tuned sim-to-real transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6939–6945.

[15] Y.-Y. Tsai *et al.*, "Droid: Minimizing the reality gap using single-shot human demonstration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3168–3175, 2021.

[16] A. Allevato, E. S. Short, M. Pryor, and A. Thomaz, "Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer," in *Conference on Robot Learning*. PMLR, 2020, pp. 445–455.

[17] F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters, "Neural posterior domain randomization," in *Conference on Robot Learning*. PMLR, 2022, pp. 1532–1542.

[18] T. Gruner, B. Belousov, F. Muratore, D. Palenicek, and J. R. Peters, "Pseudo-likelihood inference," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[19] T. Wu and J. Movellan, "Semi-parametric Gaussian process for robot system identification," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 725–731.

[20] W. Yu, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *CoRR*, vol. abs/1702.02453, 2017.

[21] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through multi-task reinforcement learning," *arXiv preprint arXiv:2302.09450*, 2023.

[22] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[23] C. E. Rasmussen and C. K. Williams, "Gaussian processes for machine learning, ser. Adaptive computation and machine learning," *Cambridge, MA, UsA: MIT Press*, vol. 38, pp. 715–719, 2006.

[24] Y. Wang, R. Gondokaryono, A. Munawar, and G. S. Fischer, "A convex optimization-based dynamic model identification package for the da Vinci Research Kit," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3657–3664, 2019.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.