Real-to-Sim via End-to-End Differentiable Simulation and Rendering

Yifan Zhu[1], Tianyi Xiang[1], Aaron Dollar[1], and Zherong Pan[2]

***Abstract*— Learning predictive world models for robots in novel environments from sparse online observations is essential for robot task planning and execution in novel environments. However, existing object representations are incapable of jointly parameterizing the shape, appearance, and physical properties of rigid objects. In this work, we introduce a novel representation that allows the system co-identification of the physical, geometry, and appearance of world models. Our method employs a novel differentiable point-based object representation coupled with a grid-based appearance field, which allows differentiable object collision detection and rendering. Combined with a differentiable physical simulator, we achieve end-to-end optimization of world models, given the sparse visual and tactile observations of a physical motion sequence. Through a series of benchmarking system identification tasks in simulated and real environments, we show that our method can learn both simulation- and rendering-ready world models from only a few partial observations.**

I. Introduction

Building a predictive world model of how a robot's actions can affect the surrounding environment is essential for planning and control. However, constructing such a model from raw observations online, in novel real-world environments remains challenging. Shapes, appearances, and physical parameters are usually partially observable, and robots are typically limited in time and computational resources.

Recently, there has been growing interest in learning world models from large offline datasets of action-labeled videos using generative modeling techniques [1], [2], [3], [4]. However, these models are susceptible to distribution shifts and cannot infer latent properties such as the coefficient of friction. In addition, they are not physically consistent and cannot provide information such as contact forces, which are essential for downstream tasks. Meanwhile, system identification of world models aims to leverage strong priors to estimate the parameters of world models from noisy, sparse, and partial observations. However, identifying general-purpose world models requires the understanding of geometry, appearance, and physical properties of the environment, such as object-segmentation labels, geometric shapes, mass-inertial properties, and frictional coefficients.

Many existing works take advantage of differentiable simulators as strong priors that allow efficient system identification of physical parameters [5], [6], [7]. Differentiable simulators allow the gradient-based optimization of mass-inertial properties and frictional coefficients, in order for a physical motion sequence to match sparse robot observations. However, these works assume known shapes and appearances of the objects involved and do not allow the algorithm to adapt the shape, appearance, and physical properties simultaneously.

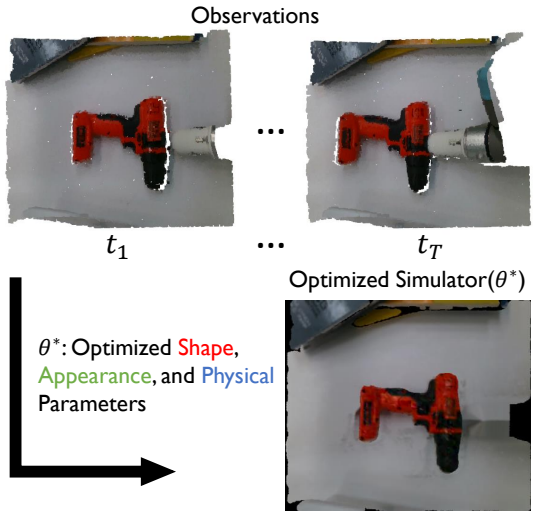

Fig. 1. Our method jointly optimizes the shape, appearance, and physical parameters of a world model in the form of a rigid body simulator, from the visual and tactile observations of a single robot push.

On the other hand, we have witnessed recent advances in learnable appearance models, such as Neural Radiance Fields (NeRF) [8] and Gaussian Splatting (GS) [9]. These methods build the rendering equation into a learnable representation to enable the identification of shapes and appearances with general geometries and topologies, from raw observations. However, rigid body simulators [10] typically require the use of volumetric representations such as convex hulls to detect collisions, while appearance models such as neural fields and Gaussian splats are incompatible with these conventional representations. In summary, to the best of the authors' knowledge, no existing method allows the simultaneous identification of physical, geometric, and appearance properties of a scene from sparse robot observations.

To tackle these challenges, this work presents an object representation that is compatible with general-purpose rigid body simulators and allows the joint optimization of geometry, appearance, and physical parameters. Our proposed representation is the combination of a recently proposed point-based shape representation Shape-as-Points (SaP) [11] and a grid-based appearance property field. SaP parame-

[1]Y. Zhu, T. Xiang, and A. Dollar are with the Department of Mechanical Engineering and Materials Science, Yale University, New Haven, United States. {yifan.zhu, tianyi.xiang, aaron.dollar}@yale.edu

[2]Z. Pan is with Lightspeed Studios, Tencent America, Bellevue, WA, United States. zrpan@global.tencent.com

terizes an object's geometry and topology using a set of surface points along with normal directions. It then uses differentiable Poisson reconstruction to recover a smooth indicator field, which can be converted to a mesh using a differentiable marching cubes algorithm. The texture of the vertices of the mesh is then obtained by interpolating the appearance grid. Employing the mesh in a differentiable rigid body simulator [12] that provides gradients for the physical parameters and contact points of the objects, our method constructs a fully differentiable pipeline for jointly optimizing the geometry, appearance, and physical parameters. To summarize, our contributions are:

- A jointly differentiable representation of objects' shape, appearance, and physical properties.
- An algorithm for identifying world models online from sparse robot observations, which we refer to as real-to-sim, with an end-to-end differentiable pipeline.

We evaluate our method on a series of benchmarking problems in both simulated and real-world environments. As illustrated in Fig. 1, the results show that our method can infer faithful world models from a single episode of robot interactions with the environment.

II. Related Work

We review related works on differentiable rigid body simulators, learnable geometry and appearance models, and identifying world models.

A. Differentiable Rigid Body Simulator

Rigid body simulators are essential tools in robotics and engineering for testing and verification, perception, control, and planning. Traditional rigid body simulators are not differentiable, but recently, there have been many attempts at building differentiable rigid body simulators [12], [13], [14], [15], [16], [17] for facilitating downstream system identification, robot planning, and policy optimization tasks. Different strategies are adopted to enable the calculation of gradients for the underlying non-differentiable contact dynamics, including employing a smooth contact model [17], [14], [15], using sub-gradients of the linear complementarity problem [13], and implicit gradients of nonlinear optimization [12], [16]. The majority of these methods, however, do not provide gradients with respect to the geometry, with the exception of [16], [12], [14]. In this work, we adopt the simulator proposed by Strecke et al. [12] for its physical realism, numerical stability, and fast computation from GPU acceleration.

B. Learnable Geometry and Appearance Models

Learning 3D geometry and appearance models from 2D raw images is vital to robots' understanding of the physical world. Earlier research has focused on learning only the 3D geometries without appearance, including point-cloud-based [18] and convex-hull-based [19] models. Neural radiance fields (NeRF) is the first method that enables a continuously learnable model for the full 3D appearance of objects and scenes, which uses neural networks to parameterize the spatial appearance properties and implicitly learn the 3D geometry. More recently, Kerbl proposed Gaussian splitting (GS) [9], a non-parametric method that represents the appearance of the scene with 3D Gaussians and significantly improves the training and rendering speeds due to their fit for fast GPU/CUDA-based rasterization. These algorithms are capable of learning detailed 3D object and scene appearances from sparse image-based observations. However, NeRF represents objects with a neural field and lacks clear definition of rigid object surfaces, and the case is similar with GS. Therefore, while there are some initial attempts at integrating them with rigid body simulators [20], [21], research for robust and physically correct collision detections with these models is still ongoing. In addition, NeRF and GS require multiple views of the image observations for supervision, which is unrealistic in typical robotics applications.

Our method is based on the shape representation SaP [11], with a set of sampled points with normals to represent the object surface. It then reconstructs the indicator field by Poisson reconstruction. A mesh can then be reconstructed from the indicator field using a differentiable marching cubes algorithm. Coupled with a 3D color grid, the mesh vertex colors can be assigned. Compared to standard 3D representations such as point clouds, which do not allow volumetric collision detection, or meshes, which do not allow topology changes during optimization, our SaP-based methods enjoy the best of both worlds. Combined with a differentiable renderer, our object representation then achieves end-to-end image-based shape optimization.

C. World Models

In this work, we interpret the term world models broadly and regard traditional system identification methods [22] as world modeling as well, where only the dynamics are identified from full state information. However, to support diverse downstream robot tasks in the real world, world models need to be built from raw observations and support both accurate dynamics prediction and photorealistic novel view synthesis. As previously mentioned, while existing works have identified world models from raw image observations using differentiable simulators, none supports simultaneous optimization of geometry, appearance, and physical parameters.

Another line of work closely related to ours is image-based generative world modeling. These works aim to predict the next RGB frame based on the current frame and action, crafted by training on diverse datasets with generative modeling techniques such as variational autoencoders [1], [2] and diffusion models [3], [4]. The key differences between our method and these works are that our simulation, grounded in physics, is always physically consistent and is a general-purpose simulator that can also provide information such as contact forces. On the other hand, purely data-driven world models poorly to novel scenarios and have limited capabilities for downstream robot tasks.

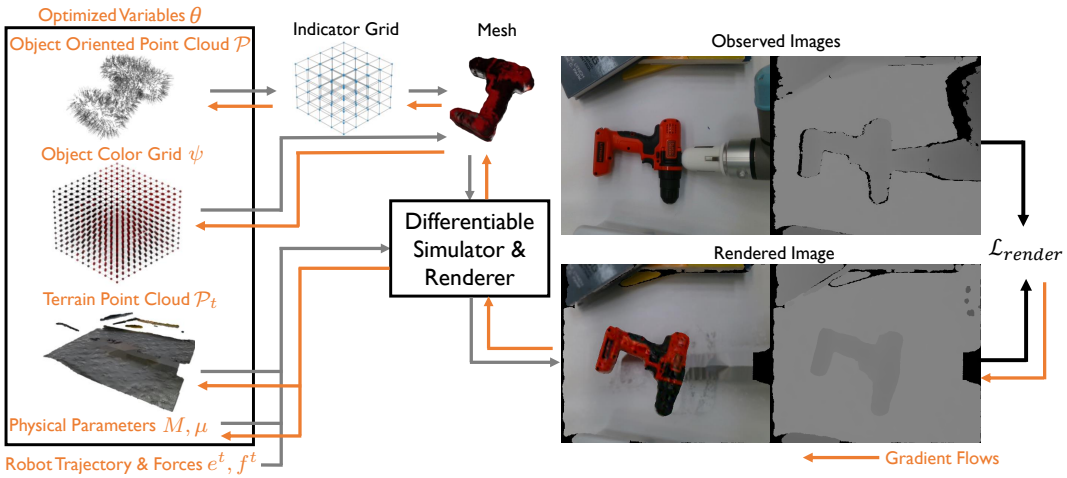

Fig. 2. Overview of the proposed fully differentiable pipeline for world model identification from sparse robot observations. Our object representation couples an oriented point cloud and a 3D color grid. Through a differentiable Poisson solver and differentiable marching cubes, the oriented point cloud is converted to an indicator grid and then a mesh, whose vertex colors are interpolated from the color grid. Feeding the object mesh, physical parameters, the terrain point cloud, and the robot pushing action into a differentiable rigid body simulator and renderer, the predicted scenes can be rendered. Calculating the loss against observed RGB-D images, the scene shape, appearance, and physical parameters are jointly optimized with gradient descent.

III. Problem Definition

In this section, we describe our formulation of world model identification, or real-to-sim. We assume the environment consists of a rigid object and a rigid terrain, whose physical properties and appearances are parameterized by θ. A robot, equipped with joint encoders and end-effector force sensors, interacts with the object at T sparse time instances: $t_1, \cdots, t_T$, with a fixed time step δt. At each time step, the robot observes its end-effector pose $e^t \in \mathrm{SE}(3)$, contact force $f^t \in \mathbb{R}^3$. Further, the robot is equipped with RGB-D camera with known intrinsics that observes the object through image $o^t \in \mathbb{R}^{H\times W\times 4}$ at camera pose $c^t \in \mathrm{SE}(3)$. We further assume an image segmentation mask $m^t \in \mathbb{R}^{H\times W\times 4}$ is provided for the object, robot, and terrain. Therefore, the robot observations are a sequence $\mathcal{O} = \{\langle t, e^t, f^t, o^t, c^t, m^t\rangle\}$, and our goal is to estimate θ from the set of sparse observations $\mathcal{O}$. We formulate this problem as a physics-constrained optimization by introducing a full-fledged physics simulator function $q^{i+1} = g(q^i, u^i, \theta)$ that can differentiate through objects' appearance, geometry, and physical parameters. Here $q^i \in \mathrm{SE}(3) \times \mathrm{SE}(3)$ is the object and robot end-effector poses at timestep t and u^i is the applied robot force to the object, which is equal in magnitude to the contact sensed contact force but opposite in direction.

Given such a simulator, the world model identification problem is formulated as solving the following optimization:

$$\begin{aligned} \underset{\theta}{\mathbf{argmin}} \sum_{t=t_1}^{t_T} \mathcal{L}(\hat{o}^t, o^t) \\ \mathbf{s.t.} \quad q^{i+1} = g(q^i, u^i, \theta) \quad \forall i = 1, \cdots, T. \end{aligned} \tag{1}$$

over a physical motion sequence of T timesteps, with the objective function $\mathcal{L}$ encourages matching the simulated observation $\hat{o}^t$ and the groundtruth observation o^t.

IV. Method

In this section, we first detail the object representation, which is key to our method. Then we describe the differentiable simulator and details on solving the optimization described in Eqn. 1.

A. Differentiable Object Representation

An ideal object representation for world model identification needs to be flexible to allow learning of complex object geometries, topologies, and appearance properties while being compatible with rigid body simulators for collision detection. Topology-agnostic geometries such as point clouds [18] and GS [9] do not allow one to calculate the penetration depth between bodies. On the other hand, meshes [14] do not allow large geometric and topological changes.

We find that the SaP framework [11], when augmented by additional appearance properties poses an ideal representation for our purpose. Briefly, this framework represents the object using a point cloud with normal on the object surface, denoted as $\mathcal{P} = \{\langle p \in \mathbb{R}^3, n \in \mathbb{R}^3\rangle\}$. These normal directions induce a discrete vector field $v(x) = \sum_{\langle p,n\rangle\in\mathcal{P}} n\mathbb{I}[x = p]$. SaP then uses Poisson reconstruction [23] to recover a so-called indicator field $\chi(x)$ and matches its gradient field with $v(x)$ by solving the variational problem:

$$\underset{\chi}{\mathbf{argmin}} \int_\Omega \|\nabla\chi(x) - v(x)\|^2,$$

which amounts to solving the Poisson equation $\Delta\chi = \nabla\dot{v}$. SaP discretizes the indicator field χ on a uniform grid domain Ω, which allows the efficition solution of χ via GPU-accelerated Fast Fourier Transform (FFT) with well-defined derivatives.

The indicator field χ is then transformed to a mesh $\mathcal{M}$ with a differentiable marching cubes algorithm [24]. Collision detection can then be easily achieved with standard techniques for meshes. To enable appearance modeling, we further augment with an appearance properties grid, with

the same grid resolution as the one storing the indicator field χ. The appearance property is then propagated to the mesh vertices via tri-linear interpolation. In this work, we only store and render the color field, denoted as ψ, but other appearance properties can be incorporated in the same manner as required by more advanced differentiable rendering equations. The mesh can then be rendered using any differentiable renderer framework such as [25], [26], for which we use the open source implementation in PyTorch3D [27]. Specifically, during the time instance t, we invoke the renderer with the object transformed to q^t and the camera transformed to c^t. In summary, our parameterization of the objects' physics and appearance is defined as:

$$\theta \triangleq \langle M(q^i), \mu, \mathcal{P}, \psi \rangle,$$

where the first two parameters are mass-inertial properties and frictional coefficients, defining the physical properties, and the last two parameters are the oriented point cloud and color field, defining the objects' appearance.

For the terrain, we simply use an oriented and colored point cloud $\mathcal{P}_t$ to represent the terrain as we do not need to simulate interactions between 2 terrains. The terrain is rendered from the colored point cloud with an alpha compositor [28] also using the PyTorch3D library and we set the radius of each point to be 0.015 m.

B. Differentiable Simulator

For our application, we only consider unconstrained rigid body dynamics with dry frictional contacts. Note that additional constraints such as soft bodies and articulated objects can be readily incorporated into our framework and its differentiation has been well-studied, e.g. in [15].

The governing equation of motion for rigid bodies and the time discretization method is well-studied and we refer the readers to Anitescu et al. [29] for details. The equation is summarized as follows:

$$M(q^i)\ddot{q}^i = C(q^i, \dot{q}^i) + J^i u^i + J^\perp \tau^\perp + J^\parallel \tau^\parallel, \tag{2}$$

with $M(q^i)$ being the generalized mass matrix, $C(q^i, \dot{q}^i)$ being the centrifugal and Coriolis force, $J^i, J_\perp, J_\parallel$ being the Jacobian matrix for the external, normal, and tangent contact forces at all the detected contact points, respectively. Finally, $\tau_\perp, \tau_\parallel$ are the contact forces. At each time step, a mixed LCP problem is solved to calculate the constraint forces $\tau_\perp, \tau_\parallel$, yielding the final acceleration $\ddot{o}^i$, and we then integrate the configuration forward in time [30], [31] as:

$$\dot{q}^{i+1} = \dot{q}^i + \ddot{q}^i \delta t \quad q^{i+1} = q^i + \dot{q}^i \delta t, \tag{3}$$

with δt being the timestep size. The mixed LCP problem is formulated as:

$$\begin{cases} 0 \le \tau_\perp & \perp J_\perp^T \dot{q}^{i+1} \ge 0 \\ 0 \le \tau_\parallel & \perp \lambda e + J_\parallel^T \dot{q}^{i+1} \ge 0 \\ 0 \le \lambda & \perp \mu \tau_\perp - e^T \tau_\parallel \ge 0, \end{cases} \tag{4}$$

with e being the unit vector and μ being the frictional coefficient. λ is an auxiliary variable deciding the static and slide frictional state. To differentiate through the simulator, we adopt the differentiation technique proposed by [13], [12], where the result of the LCP is made differentiable by solving with a primal-dual method and performing sensitivity analysis at the solution to yield derivatives with respect to the problem data. In this way, the derivatives propagates the gradient information to the Jacobian matrix $J_{\perp,\parallel}$, and finally to the object geometric data $\langle \mathcal{P}, \chi \rangle$. In summary, Eqn. 2,3,4 defines our differentiable simulator function g.

C. World Model Identification

Even with our jointly differentiable physical and appearance models, solving Eqn. 1 can still be rather difficult. This is mainly because our initial guess can be rather noisy and discrepant from the observations. As a result, the naïve gradient descent method can take many iterations and is thus prone to falling into local minima. To mitigate this issue, we propose to use two stages, and we further leverage 3D foundation models trained on web-scale data to generate good initial guesses of all the rigid objects in the scene from partial visual observations. Here we first detail our two-stage optimization formulation, and practical considerations for generating reasonable initial guesses.

1) Two-stage Optimization: We notice that the initial guess might deviate from our observations in terms of geometry, appearance, and physical parameters, the deviation in geometry and appearance can be largely corrected by considering the first static observation alone, i.e. $\langle t_1, e^{t_1}, f^{t_1}, o^{t_1}, c^{t_1}, m^{t_1} \rangle$. Therefore, our first stage considers only the first time instance and optimize θ using the following loss:

$$\begin{aligned} \mathcal{L}(\hat{o}^{t_1}, o^{t_1}) =& c_1 \mathcal{L}_{\text{rgb}}(\hat{o}^{t_1}, o^{t_1}) + c_2 \mathcal{L}_{\text{depth}}(\hat{o}^{t_1}, o^{t_1}) + \\ & c_3 \mathcal{L}_{\text{pcd}}(o^{t_1}, \theta) + c_4 \mathcal{L}_{\text{pen}}(\theta) + c_5 \sum_1^k \|q^i - q^0\|^2 + \\ & c_6 \mathcal{L}_{\text{reg}}(\theta) + c_7 \mathcal{L}_{\text{smooth}}(\theta), \end{aligned}$$

with $c_\bullet$ being the corresponding weight terms. Here $\mathcal{L}_{\text{rgb}}$ is a loss on the RGB images, defined as a weighted sum of l_1 distance and D-SSIM terms: $\mathcal{L}_{\text{rgb}} = (1-\lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{SSIM}}$, where we set $\lambda = 0.2$. $\mathcal{L}_{\text{depth}}$ is the l_1 distance on the depth images. The three loss terms $\mathcal{L}_{1,\text{SSIM,depth}}$ are further masked by m^{t_1}. $\mathcal{L}_{\text{pcd}}$ is a unilateral Chamfer distance between the observed RGB-D point cloud belonging to the object and the SaP point cloud $\mathcal{P}$, which is defined as the average of the minimum distance between each point on the observed point cloud and any point among the SaP points. The fourth and fifth terms encourage the object to achieve static force equilibrium while having no penetrations, by penalizing the object mesh penetrations in the terrain and the change in object positions over k steps, which is the sum of the positional changes over k steps by simulating forward with no robot actions. These terms provide a strong hint for the occluded part of the object. For example, when an object is lying on the table, our RGB-D observation will not cover the back side of the object. However, our model will guide SaP to fill the back-side geometries by encouraging the object to settle on the table. Finally, the last two terms regularize the object shape, where L_{reg} is the L_2 norm between the SaP points and the initial SaP points and L_{smooth} is the Laplacian smoothing objective on the object mesh.

We observe that the effects of using $\mathcal{L}_{\text{depth}}$ and $\mathcal{L}_{\text{pcd}}$ complement each other. When only $\mathcal{L}_{\text{depth}}$ is used, we cannot handle the case where a depth pixel on the ground truth image hits a mesh surface whereas the same pixel is not on the predicted image, in which case there is no gradient information for the geometry to expand and cover this pixel. Similarly, the observed point cloud used in $\mathcal{L}_{\text{pcd}}$ is based on a partial point cloud of the object, and SaP has no information about the bound of the object. To complement, $\mathcal{L}_{\text{depth}}$ informs our model of the background area and prevents the object geometries from occupying the supposed background.

After the first stage, we have tuned our model to match the first observation. When we move on to the second stage, we incorporate all timesteps by applying the robot pushing forces. For this stage, we use the simpler loss terms: $\mathcal{L} = c_8\mathcal{L}_{\text{rgb}} + c_9\mathcal{L}_{\text{pcd}} + c_{10}\mathcal{L}_{\text{robot}}$, where the first 2 terms have the same definitions as in the first stage and $\mathcal{L}_{\text{robot}}$ is the squared distance between the ground truth and predicted robot positions. During our optimization, the chain of gradient is back-propagated through the following recursive rule:

$$\frac{d\mathcal{L}(\hat{o}^{t_i}, o^{t_i})}{d\theta} = \frac{\partial \mathcal{L}}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial q^i}\left[\frac{\partial q^i}{\partial \theta} + \frac{\partial q^i}{\partial q^{i-1}}\frac{dq^{i-1}}{d\theta}\right],$$

where the first two terms $\partial\mathcal{L}/\partial\theta, \partial\mathcal{L}/\partial q^i$ is the derivatives of the rendering equation, and the remaining terms in the bracket is the derivatives of the simulator.

2) Geometry Prior: Our method relies on a reasonable initial guess. Imagine the case with an object settling on the edge of a table and the camera does not observe the contact between the two. The initial guess of the occluded part of the object could be very short and cause the object to directly fall down without touching the table. This cannot be recovered by our optimization since there are no gradients for correcting the geometries. To obtain a reasonable initial guess of the geometries and appearance of the rigid object in interest from partial visual observations, we take advantage of large reconstruction models [32] that predict object 3D models from a single RGB image, trained on web-scale data. In particular, we use TripoSR [33] in our experiments with the segmented RGB image of the object as input image. Since the generated mesh is scale- and transform-agnostic, we subsequently apply RANSAC and scale-aware iterative closest point (ICP) algorithms to register the mesh to the partial object point cloud, computed from the RGB-D image at the first time instance.

Finally, in all the experiments of this work, we assume that the occluded terrain by the object is flat, and complete the terrain by fitting a plane of points, where the colors match the nearest visible points of the terrain. In addition, in all the experiments, we do not optimize the point cloud position of the terrain and optimize only the colors. Although these settings are simplifying, we assume a similar approach could be adopted that predicts the geometry of the occluded rigid terrain from a geometry prior model and optimizes for the terrain geometry simultaneously, although more online data may be required to resolve the ambiguities of the contacts between two occluded geometries.

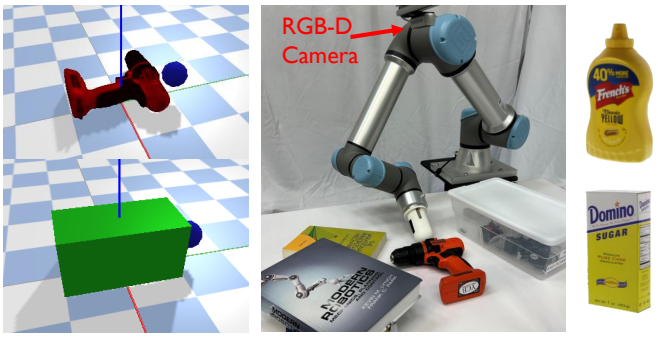

Fig. 3. The experiment setups for the simulated (left) and physical (right) experiments. Two objects (Power Drill from YCB dataset and a box) in the PyBullet simulator are used for simulation. For the real-world experiments, three YCB objects (Power Drill, Mustard Bottle, and Domino Sugar Box) are used. A UR5e arm equipped with a pusher and an overhead Realsense D435 RGB-D camera are used.

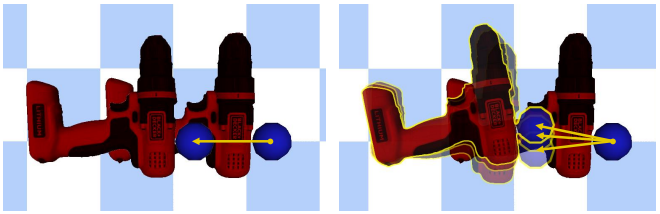

Fig. 4. The train (left) and 3 different test (right) trajectories of the `Drill` object during stage 2 in simulation, with the first and last frames shown. The blue sphere represents a floating robot. The train and testing trajectories are the same for the other objects in both simulated and physical experiments.

V. Experiments and Results

To validate our method, we first conduct experiments with simulated data, and then in the real world.

A. Simulated Experiments

Shown in Fig. 3, we conduct all the simulated experiments using data collected with the PyBullet simulator [34]. We use a simple green box object (`Box`) and a power drill object (`drill`) from the YCB object dataset [35]. The objects are placed on a flat surface with checker patterns and pushed by a floating sphere robot, while a static overhead camera takes pictures. The optimized simulator parameters are then tested on three different pushes. The pushing trajectories are visualized in Fig. 4 where they are the same for both objects. All the trajectories have $T = 30$ time steps with $\delta t = 0.01$ s. We use the following weights for optimization: $[c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}] = [10, 5000, 500, 1, 1000, 100, 4000, 10, 500, 100]$. These terms are not carefully tuned and are set such that each term has a similar order of magnitudes at the start of optimization for our experiments. In addition, we set $k = 3$ steps for simulating forward with no actions in Stage 1.

The physical parameter we optimize for is the coefficient of friction between the object and the terrain. We assume the mass is known here since optimizing both the mass and

	Novel View Synthesis			Train Dynamics Error				Test Dynamics Error		
Object	MSE ↓	SSIM ↑	PSNR ↑	Unilateral Chamfer (mm)	Pos. (mm)	Rot. (°)	μ	Unilateral Chamfer (mm)	Pos. (mm)	Rot. (°)
Box	0.00312	0.950	25.1	1.99	12.2	0.749	0.107	3.50	11.6	23.3
Drill	0.00413	0.942	24.0	4.91	15.1	3.85	0.0970	8.21	12.3	12.7

TABLE I

NOVEL VIEW SYNTHESIS AND DYNAMICS PREDICTION ERRORS FOR THE SIMULATED EXPERIMENTS

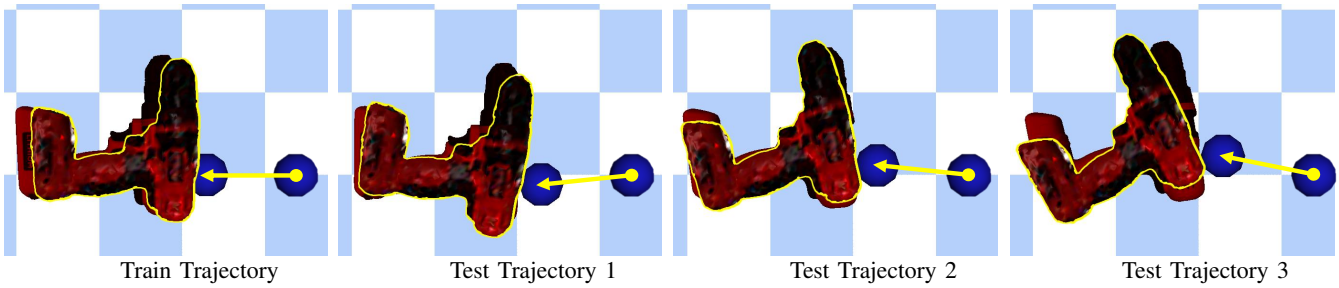

Fig. 5. The predicted and ground truth poses of the Drill object at the last frame of the train and the 3 different test trajectories in one optimization trial. The predicted object poses with the optimized θ highlighted with a yellow silhouette are overlaid with the ground-truth object, robot, and background.

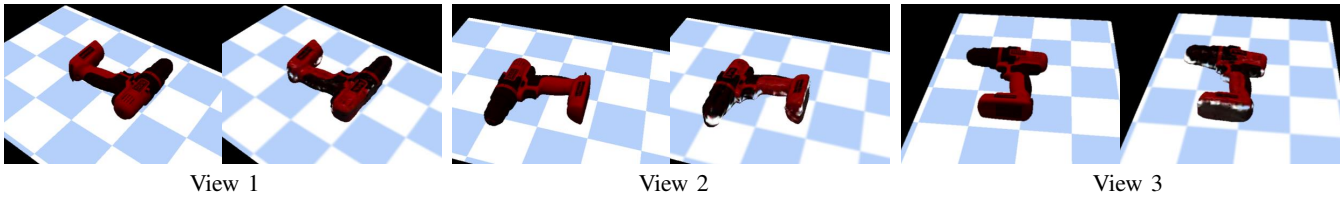

Fig. 6. The ground truth (left) and predicted (right) RGB images of 3 novel views of the Drill in simulation. Because the optimized mesh shape is slightly larger than the ground truth, the optimized color (with an overhead view during training) at some locations on the mesh has the color of the terrain. The terrain checkers are not as sharp as the ground truth due to the use of point rendering of the colored terrain point cloud.

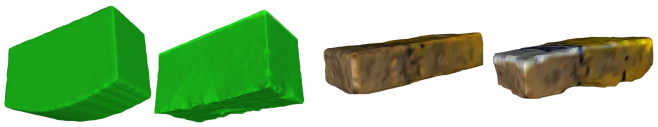

Fig. 7. The initial guess and optimized shape of the Box object in simulated experiments and the Sugar object in physical experiments during stage 1. The initial guess of the Box with the bottom bulges out would penetrate the terrain underneath. On the other hand, the initial mesh of Sugar is too thin and does not touch the terrain. Our algorithm is able to optimize the shape so that it satisfies the physics, albeit not perfect since the objects are completely occluded at the bottom.

Object	Train Unilateral Chamfer (mm)	Test Unilateral Chamfer (mm)
Drill	5.22	7.10
Mustard	3.56	6.16
Sugar	4.16	5.26

TABLE II

DYNAMICS PREDICTION ERRORS FOR THE REAL WORLD EXPERIMENTS

the coefficient of friction with a pushing trajectory is ill-conditioned. However, the mass can be easily identified with some prehensile trajectory like picking up the object. We also assume that the center of mass is at the geometry center of the initial geometry guess and that the inertia matrix is calculated assuming the object is a box with the dimensions of the object bounding box. We note that while we make these simplifying assumptions, this is not a limitation of our method and these physical parameters can all be optimized for in the fully differentiable pipeline with more online data.

We report the quantitative results for both objects in Table I, where we show the quality of novel view synthesis and the errors in predicting the dynamics. For each object, we run three optimization trials with different random initial coefficients of friction between 0.1 and 1, and the average across the trials are reported in Table I. We also show the final location of the pushed objects in both the training and testing pushing trajectories in Fig. 5 and some examples of novel view synthesis in Fig. 6. Our method identifies the shape and appearance parameters, and coefficient of friction that lead to accurate dynamics prediction and high-quality novel view synthesis. We think the error in the coefficient of friction could be due to the inconsistencies between the physics engine of our method and PyBullet. Due to the slight mismatching between the optimized and ground truth shape of the object geometry, parts of the sides of the optimized Drill mesh have the color of the terrain. We believe this could be resolved with slightly more diverse angles other than a static overhead camera. For the Box object, instead of using an initial geometry guess generated by a reconstruction model, we use an arbitrarily created box that clearly penetrates the terrain underneath to demonstrate the ability of our method to adjust occluded geometry based on the physics, shown in Fig. 7.

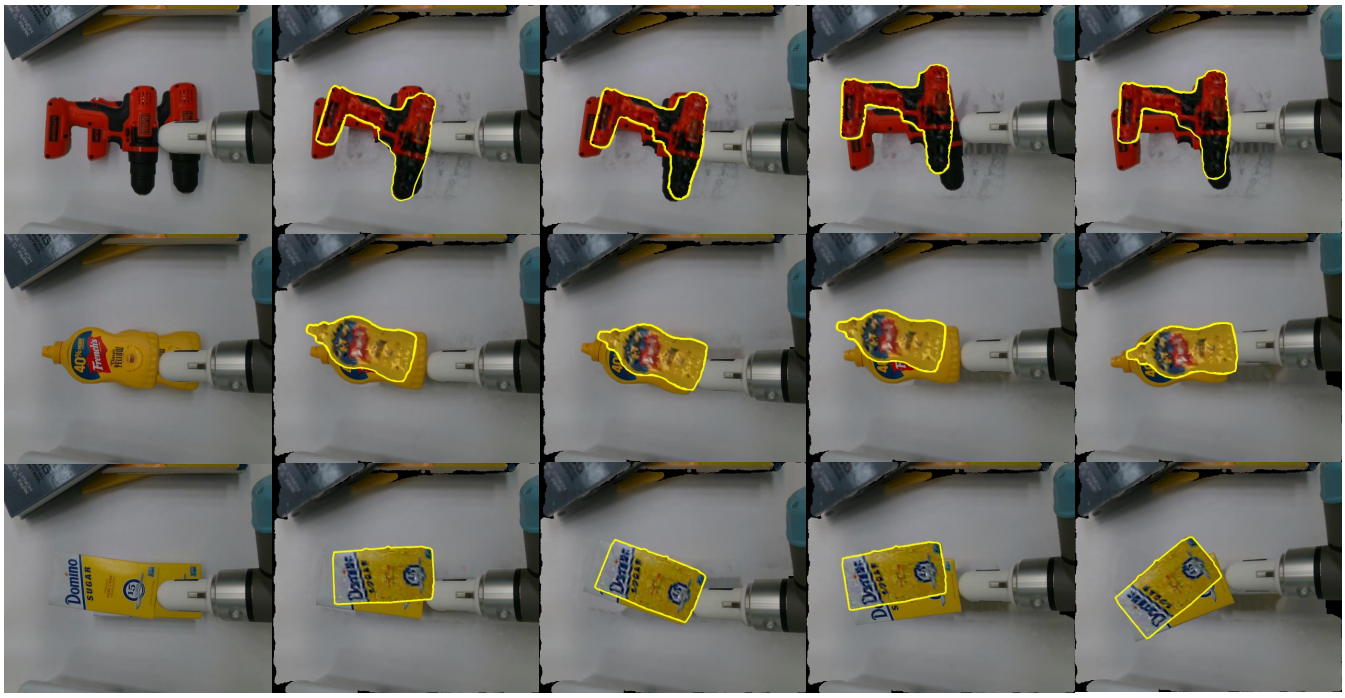

Fig. 8. Train and test results of the physical experiments. Each row shows the results from an object. The columns from left to right are: 1) the robot's push to collect the training data, 2) the optimized object position at the end of the train trajectory, and 3)-5) those at the end of the three different test trajectories. The predicted object poses with the optimized θ highlighted with a yellow silhouette are overlaid with the ground truth object and robot, and the background rendered from the optimized simulator.

B. Physical Experiments

Shown in Fig. 3, we conduct physical experiments with a UR5e robot arm equipped with an embedded end-effector F/T sensor and a pusher, and a static overhead Realsense D435 RGB-D camera. We adopt the same train and test pushing trajectories as the simulated experiments for 3 YCB objects: power drill (`Drill`), sugar box (`Sugar`), and mustard bottle (`Mustard`). However, we use a total of $T = 48$ time steps with $\delta t = 0.03\,\text{s}$.

The dynamics prediction results are reported in Table II, and some examples of pushing trajectories are visualized in Fig. 8. For each object, we use 0.4 as the initial guess of the coefficient of friction for optimization. Note that because we do not have access to the ground truth object poses and coefficients of friction, we only report the unilateral chamfer distances. Overall, the errors are comparable to those from the simulated experiments. In addition, we show the initial and optimized shape of the `Sugar` object in Fig. 7, where the initial shape is too thin and does not contact the terrain below. Our algorithm modifies the occluded geometry to satisfy the physics. While the occluded geometry is not perfect, our method allows the object geometry to be further improved if given more diverse online data such as flipping the object.

VI. Limitations

Our method assumes ground-truth object masks from the scene, which might not always be possible even with advanced foundational segmentation models. In addition, our method does not consider more advanced appearance models, lighting sources, and shadows. As a result, the rendered scenes could have artifacts that do not match the real-world observations. Currently, each optimization run is completed in under 15 mins on a standard PC with an Intel i9-13900KF CPU, 64 GB of RAM, and a GeForce RTX 4090 GPU. While this is not ideal for online robotics applications, we intend to reduce the runtime by using better initial guesses of geometry and physical parameters from data-driven pre-trained models and more efficient implementation.

VII. Conclusion

We propose a novel algorithm to solve the system co-identification task of objects' physical properties and well as appearance models, a crucial step in almost all the downstream robot manipulation tasks. To the best of our knowledge, this is the first method that allows the joint optimization of both properties. Our method combines the merit of SaP object representation [24], differentiable collision detection [36], and differentiable simulation [12]. Although our method has several limitations, it opens doors to a rich spectrum of future research topics. Some potential future directions include extending our method to identify multi-body dynamic systems with additional constraints, more advanced appearance models, and active perception to correct for wrong object masks.

References

[1] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[2] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps *et al.*, "Genie: Generative interactive environments," 2024.

[3] F. Zhu, H. Wu, S. Guo, Y. Liu, C. Cheang, and T. Kong, "Irasim: Learning interactive real-robot action simulators," *arXiv preprint arXiv:2406.14540*, 2024.

[4] M. Yang, Y. Du, K. Ghasemipour, J. Tompson, L. Kaelbling, D. Schuurmans, P. Abbeel, U. Berkeley, and G. DeepMind, "Learning interactive real-world simulators," 10 2023. [Online]. Available: https://arxiv.org/abs/2310.06114v2

[5] C. Song and A. Boularias, "Learning to Slide Unknown Objects with Differentiable Physics Simulations," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[6] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, 2021.

[7] K. Ehsani, S. Tulsiani, S. Gupta, A. Farhadi, and A. Gupta, "Use the force, luke! learning to predict physical forces by simulating effects," 2020.

[8] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[9] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[10] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 4397–4404.

[11] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable poisson solver," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 032–13 044, 2021.

[12] M. Strecke and J. Stueckler, "DiffSDFSim: Differentiable rigid-body dynamics with implicit shapes," in *International Conference on 3D Vision (3DV)*, Dec. 2021.

[13] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and Feature-Complete Differentiable Physics Engine for Articulated Rigid Bodies with Contact Constraints," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[14] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal, "An End-to-End Differentiable Framework for Contact-Aware Robot Design," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[15] Y. Qiao, J. Liang, V. Koltun, and M. Lin, "Differentiable simulation of soft multi-body systems," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 17 123–17 135. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/8e296a067a37563370ded05f5a3bf3ec-Paper.pdf

[16] T. A. Howell, S. L. Cleac'h, J. Brüdigam, J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable physics engine for robotics," *arXiv preprint arXiv:2203.00806*, 2022.

[17] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: Analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[18] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*. PMLR, 2018, pp. 40–49.

[19] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi, "Cvxnet: Learnable convex decomposition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 31–44.

[20] N. Sharp and A. Jacobson, "Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–16, 2022.

[21] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang, "Physgaussian: Physics-integrated 3d gaussians for generative dynamics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4389–4398.

[22] K. J. Åström and P. Eykhoff, "System identification—a survey," *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[23] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, no. 4, 2006.

[24] Y. Liao, S. Donne, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2916–2925.

[25] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3d reasoning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7708–7717.

[26] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila, "Modular primitives for high-performance differentiable rendering," *ACM Transactions on Graphics*, vol. 39, no. 6, 2020.

[27] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3d deep learning with pytorch3d," *arXiv:2007.08501*, 2020.

[28] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "Synsin: End-to-end view synthesis from a single image," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7467–7477.

[29] M. Anitescu and F. A. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, pp. 231–247, 1997.

[30] M. B. Cline, "Rigid body simulation with contact and constraints," Ph.D. dissertation, University of British Columbia, 2002.

[31] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[32] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan, "Lrm: Large reconstruction model for single image to 3d," *arXiv preprint arXiv:2311.04400*, 2023.

[33] D. Tochilkin, D. Pankratz, Z. Liu, Z. Huang, , A. Letts, Y. Li, D. Liang, C. Laforte, V. Jampani, and Y.-P. Cao, "Triposr: Fast 3d object reconstruction from a single image," *arXiv preprint arXiv:2403.02151*, 2024.

[34] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[35] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics and Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[36] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," *ACM transactions on graphics (TOG)*, vol. 22, no. 3, pp. 871–878, 2003.