Data-Augmented Contact Model for Rigid Body Simulation

Yifeng Jiang, Jiazheng Sun, and C. Karen Liu
Georgia Institute of Technology

Abstract—Accurately modeling contact behaviors for real-world, near-rigid materials remains a grand challenge for existing rigid-body physics simulators. This paper introduces a data-augmented contact model that incorporates analytical solutions with observed data to predict the 3D contact impulse which could result in rigid bodies bouncing, sliding or spinning in all directions. Our method enhances the expressiveness of the standard Coulomb contact model by learning the contact behaviors from the observed data, while preserving the fundamental contact constraints whenever possible. For example, a classifier is trained to approximate the transitions between static and dynamic frictions, while non-penetration constraint during collision is enforced analytically. Our method computes the aggregated effect of contact for the entire rigid body, instead of predicting the contact force for each contact point individually, removing the exponential decline in accuracy as the number of contact points increases.

I. INTRODUCTION

The reality gap, as termed in the robotics community [1], between virtual simulations and the physical world renders control policies developed primarily from simulations ineffective in real life scenarios. The accuracy of contact modeling is one of the most important factors that give rise to the reality gap, especially to humanoid locomotion or manipulation in contact-rich environments. Most existing simulators use an idealized Coulomb friction model, an empirical construct to approximate the changes between two physical regimes (static friction vs dynamic friction). The Coulomb friction model assumes linear relationship between normal force and frictional force, using a single friction coefficient to represent an isometric friction cone. In addition, the computation of contact force also involves approximation and arbitrary decisions. Many existing simulators formulate a Linear Complementarity Program (LCP), which solutions are not unique except for the frictionless case [2]. Depending on the initial guesses and numerical methods used for solving the LCP problem (e.g. Lemke method [3] vs Gauss-Seidel algorithm [4]), the resulting contact forces can be drastically different. These existing issues suggest that an computational contact model grounded by real-world observations can be a desired alternative.

While recent work has shown that physical phenomena can be learned from data and approximated by neural networks with vision perception as input, precisely enforcing constraints, such as contacts, remains difficult for these function approximators learned in an end-to-end fashion. Consider a box resting on a table. If the contact force is slightly larger or smaller than the gravitational force, we will start to see the box rattling or sinking into the table without any external force. Such categorically incorrect prediction of physics outcomes are likely to have negative impacts on the development of control policies.

This paper introduces a data-augmented contact model that incorporates analytical solutions with empirical data collected for a particular scenario (e.g. a specific robot foot contacting a specific surface), such that the simulated results better match the observed phenomenon.

Our approach is built on two key insights. First, we utilize analytical solutions from first principles whenever possible and only resort to data-driven approach when the phenomenon is less well understood. For example, the contact force that prevents the objects from interpenetrating is enforced by equations instead of learned from the data. In contrast, we rely on real-world observations to model and validate the less-understood static friction condition. To this end, we propose to decompose the contact problem to two steps: predicting the next contact state (i.e. static, dynamic, or detach) and determining contact forces. We solve the first step by learning a classifier from the observed data and the second step by a combination of learning a regressor and solving constrained systems. Second, we propose to compute the aggregated effect of contact at the rigid-body level, instead of predicting the contact force at each individual contact point. This decision removes the concern of the exponential decline in the prediction accuracy as the number of contact points increases.

Our approach can be used on articulated systems. Once a rigid body’s contact model is trained, we can simply connect the rigid body to an articulated system without retraining the contact model. This implies that the collision data for training the contact model can be collected using replicas of the disassembled end-effector, without putting the entire robotic system at risk.

We envision that our method will be used in the real world where learning the contact behaviors of a specific robot part (e.g. the foot or the hand) and a specific surface is crucial for the controller design. Because our work focuses on a full 3D scenario in which the object can bounce, slide, or spin in all directions, existing contact datasets, such as MIT Push [5] are not suitable for our evaluation. Instead, we learn the data-driven models from simulated collision data. Evaluations show that the data-augmented model matches contact behaviors in simulation better than a purely statistical model. Since our method is agnostic to

Yifeng Jiang, Jiazheng Sun, and C. Karen Liu are with School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA, yjiang340@gatech.edu
the choice of simulator that generates training data, it serves as a first step toward a computationally tractable 3D contact model capable of predicting the behaviors of real-world, near-rigid materials.

II. RELATED WORK

Contact and friction is a common but extremely complex phenomenon, which continuously fascinates generations of scientists and engineers. Since Coulomb and Amontons in the 18th century made the distinction between static and dynamic frictions [6], inadequacies and controversies of Coulomb’s law have been extensively studied. For example, Oden and Martins [7] summarized that friction could depend on normal force or stress, normal separation distance, slip displacement, slip velocity, time of stationary contact, slip history, and vibrations. As such, many empirical models that substitute Coulomb’s law have been proposed and are summarized in [8]. There is no model that is more accurate than the others in all scenarios. Goyal et al. [9] proposed the concept of limit surface to enclose all possible friction forces on an object during planer sliding. Similar to their work, our model works on the rigid-body level instead of point contact level, and looses the assumptions that friction must oppose the direction of motion and that friction is isotropic.

In computer animation and robotics, the simplest Coulomb’s law and the point contact representation are often used to approximate contact physics for visualizing or evaluating robotic algorithms. Existing rigid-body simulators often solve a Linear Complementarity Problem (LCP) by enforcing unilateral constraints and complementarity between relative velocities and contact forces [10], [11], [12]. In contrast, Todorov [13] relaxed contact constraints to solve a convex optimization problem at each time step. Constraint solving methods can be formulated based on force or impulse. The former enforces constraints on acceleration level [10] while the latter enforces constraints implicitly on velocity level [14].

Alternatively, data-driven methods have been applied to approximate physics laws, especially when precise physical models are unavailable or computationally intractable. Grzeszczuk et al. [15] trained neural networks to learn the system transition dynamics for generating animations. Hsu and Keyser [16] replaced the collision handling stage with statistical approximators to accelerate large-scale rigid-body animation. More recently, machine learning methods are used to accelerate fluid simulation [17], [18], [19]. The focus of these approaches has been on the performance of simulation, while our work aims to utilize the expressiveness of statistical models to increase physical fidelity of contact behaviors. Recently, researchers started to investigate the possibility of teaching neural networks physical intuitions from vision perception in an end-to-end fashion [20], [21], [22]. In contrast, our work enhances the expressiveness of the well studied contact models by learning contact behaviors from observed data, while preserving fundamental contact constraints through analytical expressions.

With the emergence of deep reinforcement learning, developing physics simulators capable of predicting the real world has been actively studied. Similar to our work, Zhou et al. [23] built a data-efficient model for the limit surface of an object during planer sliding. Bauza and Rodriguez [24] built a probabilistic model for planer sliding that takes into account the stochasticity of frictional forces. Fazeli et al. [25] proposed either to train a purely data-driven model, or to use data to learn the optimal parameters of an analytical model for planer impact. The advantages of integrating neural networks with analytical models have been demonstrated recently. Ajay et al. [26] introduced a method that trains a recurrent neural network to predict the deviation between real-world contact trajectories and those computed by a physics engine, while Kloss et al. [27] proposed to combine a neural network for perception with a physics model for prediction. Both work demonstrated the advantages of hybrid methods using a planar pushing example. Instead of treating the physics engine as a ”black box” and correcting its output, our work directly improves the physics engine using learned function approximators. We also show that, by predicting contact impulses instead of the entire state of the system, we can reuse the learned contact models in new dynamic systems without retraining.

III. METHOD

We propose a method to predict the contact impulse between a specific pair of near-rigid objects. An ideal contact model in a physics simulator should at least guarantee the following properties:

1) Non-penetration: The geometries of the objects in contact should not overlap.
2) Repulsive force: The contact force should only push the objects away instead of pulling them together.
3) Workless condition: The contact force becomes zero at the instance when the bodies begin to separate.
4) Two friction regimes: There is an unsmooth switch between static and dynamic friction forces, depending on the materials of the objects and other factors.
5) Dynamic friction model: The dynamic friction force depends on the normal contact force, the relative velocity of the objects, and other factors.

Among these five properties, (4) and (5) depend on empirical models because their mechanics are not well understood. As such, our method will use a data-driven approach to achieve (4) and (5), while maintaining the analytical solution that satisfies (1)-(3).

A. Assumptions

We address a perfectly inelastic (i.e. the coefficient of restitution is zero), non-adhesive contact phenomenon between two near-rigid objects, which is a common scenario in robotic applications. We assume both objects have convex shapes and that one of them is stationary. The deformation during the collision is negligible comparing to the overall rigid body motion. We assume the time is discretized and that the applied forces (i.e. all forces other than the constraint
force) are integrated prior to solving the constraint force (see more discussion in Section \( \sqrt{3} \)). We expect the range of collision impulses encountered at test time to lie within the range used for training. For collision between articulated rigid body systems, we assume that the distal link is the only part of the system that is in contact.

We compute the aggregated effect of contact for the entire rigid body, instead of using point-contact representation, which increases computational complexity and leads to issues with over-parameterization. Our method represents the contact geometry as a patch with non-zero area on the surface of the object, denoted as \( \mathcal{P} \). We assume that the normal direction of the contact patch is well-defined.

### B. Single rigid body

The contact computation in physics engines typically consists of two separate processes, contact detection that identifies the contact location on the surface of the object, and contact handling that calculates the contact force such that the contact constraints and the equations of motion are satisfied. Using a standard contact detector, \( D(\mathbf{q}_t) \), we can compute a contact patch \( \mathcal{P} \), represented by the convex hull of contact points, given shape and current position of the rigid body \( \mathbf{q}_t \in \mathbb{R}^6 \) expressed in the generalized coordinates.

The main challenge of contact computation lies in contact handling. In its most general form, a contact handling routine \( \mathbf{q}_{t+1} \) maps the pre-contact state (i.e. \( \mathbf{q}_t \)) can be expressed as a function \( \mathbf{q}_{t+1} = H(\mathbf{q}_t, \mathbf{q}_t, \mathbf{\tau}_t, \mathcal{P}) \), which maps the pre-contact state (i.e. \( \mathbf{q}_t \) and \( \mathbf{q}_t \)), applied forces \( \mathbf{\tau}_t \), and contact patch \( \mathcal{P} \) to the 6D contact impulse, \( \mathbf{p} \). Precisely, \( \mathbf{p} \) is the integrated pressure on the contacting surface over the entire contact patch \( \mathcal{P} \) and over the time step interval \( h \). During the collision process, the contact pressure might not be constant, but its aggregated effect in one time step is equivalent to the impulse \( \mathbf{p} \), which can be used to integrate the state forward to the next time step by

\[
\mathbf{q}_{t+1} = \mathbf{M}(\mathbf{q}_t)^{-1}(h \mathbf{\tau} + \mathbf{p}),
\]

and

\[
\mathbf{q}_{t+1} = \text{Integrate}(\mathbf{q}_t, \mathbf{q}_{t+1}),
\]

where \( \mathbf{M}(\mathbf{q}_t) \) is the generalized mass matrix of the rigid body. At every time step, if \( D \) detects a non-empty \( \mathcal{P} \), we invoke the contact handler \( H \), described in Algorithm 1.

We first update the current velocity \( \mathbf{q}_t \) to an intermediate velocity by explicitly integrating the applied force \( \mathbf{\tau}_t \): \( \mathbf{q}^{(1)} = \mathbf{q}_t + h \mathbf{M}^{-1} \mathbf{\tau} \). Directly training a regressor to predict the contact impulse \( \mathbf{p} \) is likely to violate Properties (1)-(3), as they require precise satisfaction of constraints. Instead, we use observed data to train a classifier \( C(\mathbf{q}_t, \mathbf{q}^{(1)}) \) that predicts one of the following outcomes for \( \mathcal{P} \): static, dynamic, or detach. Based on the predicted outcome, we will calculate the contact impulse differently in order to satisfy (1)-(3).

**a) Static case:** The static case indicates that \( \mathcal{P} \) will remain in the same position and orientation at the beginning of next time step, but the rigid body is not necessarily stationary. Fig. 1 illustrates an example in which the rigid body is moving while \( \mathcal{P} \) is static. The contact impulse in this case must ensure that \( \mathcal{P} \) has zero velocity at the end of this time step. In physics engines, this is easily achieved by setting positional constraints at \( \mathcal{P} \) and solve for the constraint impulse \( \mathbf{p} \) that satisfies \( \mathbf{v}_p = 0 \), where \( \mathbf{v}_p \) is the generalized velocity of \( \mathcal{P} \) at the end of time step. Since \( \mathbf{v}_p \) and \( \mathbf{p} \) have the same effective degrees of freedom, the linear system \( \mathbf{v}_p(\mathbf{p}) = 0 \) has a unique solution. Therefore, if a state is classified as "static" by a highly accurate classifier \( C \), the solution of \( \mathbf{v}_p(\mathbf{p}) = 0 \) will satisfy Properties (1)-(3).

**b) Dynamic case:** In the dynamic case, the contact patch \( \mathcal{P} \) will change its position or orientation, or lose some area at the next time step (Fig. 1). To compute the motion of \( \mathcal{P} \), we need to predict the contact impulse \( \mathbf{p} \). We propose to train a regressor from the observed data because the idealized Coulomb friction model is limited when approximating the complex dynamic friction phenomena. However, directly using a regressor to predict \( \mathbf{p} \) will still suffer from the same problem of failing to satisfy Properties (1)-(3) precisely.

Instead, our algorithm first reparameterizes and decouples \( \mathbf{p} = (p_x, p_y, p_z, m_x, m_y, m_z)^T \), where \( p \) and \( m \) indicate linear impulse and impulsive torque respectively, into a frictional impulse \( \mathbf{p}_f = (p_x, p_y, m_y)^T \) and a normal impulse \( \mathbf{p}_n = (m_x, \tilde{m}_x, \tilde{m}_z)^T \). We define \( \tilde{m}_x \) and \( \tilde{m}_z \) to be the impulsive torques induced by the normal linear impulse \( p_y \). With this decoupling, in a perfectly inelastic case (i.e. restitution is zero), given any \( \mathbf{p}_f \), there exists one unique

---

**Algorithm 1 Single body contact model: \( H \)**

**Input:** \( \mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{\tau}_t, \mathcal{P} \)

**Output:** \( \mathbf{p} \)

1: \( \dot{\mathbf{q}}^{(1)} = \dot{\mathbf{q}}_t + h \mathbf{M}^{-1} \mathbf{\tau} \)
2: \( \mathbf{c} \leftarrow C(\mathbf{q}_t, \dot{\mathbf{q}}^{(1)}) \)
3: if \( \mathbf{c} == \text{"static"} \) then
4: \( \text{AddPositionConstraint}() \)
5: \( \mathbf{p} \leftarrow \text{SolveConstraint}(\mathbf{q}_t, \dot{\mathbf{q}}^{(1)}) \)
6: \( \text{RemovePositionConstraint}() \)
7: else if \( \mathbf{c} == \text{"dynamic"} \) then
8: \( \mathbf{p}_f \leftarrow R(\mathbf{q}_t, \dot{\mathbf{q}}^{(1)}) \)
9: \( \dot{\mathbf{q}}^{(2)} = \dot{\mathbf{q}}^{(1)} + M^{-1} \mathbf{\tau}_f \)
10: \( \mathbf{p}_n \leftarrow \text{FrictionlessLCP}(\mathbf{q}_t, \dot{\mathbf{q}}^{(2)}) \)
11: \( \mathbf{p} = \mathbf{T}_n \mathbf{p}_n + \mathbf{T}_f \mathbf{p}_f \)
12: else
13: \( \mathbf{p} = 0 \)
14: return \( \mathbf{p} \)

---

**Fig. 1.** Illustration of "static" and "dynamic" cases. The classifier predicts "static" at \( t_n \) and \( t_{n+1} \). We analytically solve a contact impulse to ensure the contact patch (shown in red) has zero velocity at the end of \( t_n \) and \( t_{n+1} \). The classifier predicts "dynamic" at \( t_{n+2} \) and expects the contact patch to change at the end of \( t_{n+2} \).
p_n such that Properties (1)-(3) are satisfied. Therefore, we train a regressor to only predict p_f and analytically calculate the unique solution for p_n based on the predicted p_f. Specifically, Algorithm 1 in the dynamic case first predicts p_f using the trained regressor \( R(q_t, \dot{q}^{(1)}) \) (Line 8) and then integrates p_f to obtain a second intermediate velocity: \( \dot{q}^{(2)} = \dot{q}^{(1)} + M^{-1}T_f p_f \) (Line 9). Here \( T_f \in \mathbb{R}^{6 \times 3} \) transforms p_f to the generalized coordinates. The unique solution for p_n can be solved by any routine that respects normal complementaries. Our algorithm uses a Danzig-like positive definite LCP solver for frictionless contacts [29] (Line 10). Finally, we combine the analytical p_n and predicted p_f to obtain p in generalized coordinates (Line 11).

If the regressor \( R \) were perfectly accurate, the uniqueness of \( p_n \) ensures that the decoupling treatment in Algorithm 2 does not affect the true solution of the contact impulse p. When \( R \) is not perfectly accurate, the frictionless LCP (Line 10) serves as a corrective step on p that prioritizes the satisfaction of Properties (1)-(3) over Property (5).

c) Detach case: In the detach case, \( \mathcal{P} \) is predicted to have positive normal velocity and leave the surface at the next time step. Therefore, given zero restitution, we set \( p = 0 \), ensuring that Property (3) is satisfied.

**Remarks:** Our method addresses Property (4) by learning a classifier from the observed data. Similarly, the regressor incorporates the observed data to address Property (5). Assuming that the classifier is highly accurate, the resultant contact impulse for the static and detach cases will closely match reality and satisfy Properties (1)-(3) exactly. The classifier will be less accurate near the decision boundary, which coincides with the poorly-understood region where transitions between different physical regimes occur.

### C. Articulated rigid bodies

Our method can be extended to articulated rigid body systems. We decompose the state of the system into the distal body \((q_t, \dot{q}_t)\) and all other bodies in the upstream system \((\dot{q}, \dot{q})\) (Fig. 2). The joint force \( f_j \) transmitted between the distal body and the upstream system is unknown and must be solved simultaneously with the contact impulse \( p \).

![Fig. 2. An articulated rigid body system which consists of a distal body whose state is \((q, \dot{q})\) and other bodies in the upstream system \((\dot{q}, \dot{q})\). We solve for joint constraint force \( f_j \) and contact impulse \( p \) simultaneously.](image)

The actual distribution of normal force over the contact patch is still undetermined. [28]

---

**Algorithm 2** Articulated bodies contact solver

**Input:** \((q_t, \dot{q}_t, \tau), (\dot{q}_t, \dot{q}_t, \tau), \mathcal{P}\)

**Output:** \((q_{t+1}, q_{t+1}, \dot{q}_{t+1}, q_{t+1}, \dot{q}_{t+1})\)

1. Initialize \( f_j \) using Eq. (7)
2. Calculate \( J, M, J, M \)
3. while solver not terminated do
   4. Evaluate \( G(f_j) \) using Eq. (6)
   5. Update \( f_j \) according to Powell’s method
   6. \( p \leftarrow H(q_t, \dot{q}_t, \tau + J^T f_j, \mathcal{P}) \)
   7. \( \dot{q}_{t+1} = \dot{q}_t + hM^{-1} \tau + hM^{-1} J^T f_j + M^{-1} p \)
   8. \( \dot{q}_{t+1} = \text{Integrate}(q_t, \dot{q}_{t+1}) \)
   9. \( \dot{q}_{t+1} = \dot{q}_t + hM^{-1} \tau - hM^{-1} J^T f_j \)
   10. \( \dot{q}_{t+1} = \text{Integrate}(q_t, \dot{q}_{t+1}) \)
11. return \( q_{t+1}, \dot{q}_{t+1}, \dot{q}_{t+1}, \dot{q}_{t+1}, \dot{q}_{t+1} \)

Algorithm 2 starts with expressing the velocity of the distal body at the next time step:

\[
\dot{q}_{t+1} = \dot{q}_t + hM^{-1} \tau + hM^{-1} J^T f_j + M^{-1} p, \tag{3}
\]

where \( M(q_t) \) is the generalized mass matrix for the distal body and \( J(q_t) \) is the Jacobian transforming from the generalized coordinates of the rigid body to the Cartesian space at the joint. The Cartesian velocity at the joint at the next time step is then given by

\[
\dot{v}_{t+1} = J\dot{q}_t + hJM^{-1} \tau + hJM^{-1} J^T f_j + JM^{-1} p. \tag{4}
\]

Similarly, the velocity of the upstream system evaluated at the joint can be expressed as

\[
\dot{v}_{t+1} = J\dot{q}_t + hJM^{-1} \dot{\tau} - hJM^{-1} \dot{\bar{J}}^T f_j, \tag{5}
\]

where \( M \) and \( J \) are the mass matrix and Jacobian for the upstream system.

Since the joint constraint is satisfied at the beginning of the time step \( t_0 \), we only need to ensure that the velocity of the constraint is satisfied so that at \( t_1 \) the distal body and the upstream system still coincide at the joint. Therefore, we need to solve for a \( f_j \) such that \( \dot{v}_{t+1} - \dot{v}_{t+1} = 0 \):

\[
G(f_j) = \dot{v}_{t+1} - \dot{v}_{t+1} = Af_j + JM^{-1} H(f_j) + c = 0, \tag{6}
\]

where \( A = hJM^{-1} J + hJM^{-1} \dot{\bar{J}} \) and \( c = J\dot{q}_t + hJM^{-1} \dot{\tau} - (J\dot{q}_t + hJM^{-1} \dot{\bar{J}}) \) are constants in the equation given \( q_t \) and \( \dot{q}_t \). \( H(f_j) \) is a shorthand for \( H(q_t, \dot{q}_t, \tau + J^T f_j, \mathcal{P}) \), which outputs \( p \) depending on \( f_j \).

We solve Eq. (6) using Powell hybrid method [30], which uses finite difference to approximate the Jacobian matrix and is less sensitive to the initial guess to the problem. Powell’s method only requires a routine to evaluate \( G(f_j) \) and an initial guess. Using the heuristic that assumes \( \dot{v}_{t+1} = 0 \), we compute the initial \( f_j \) by

\[
f_j = (hJM^{-1} J + hJM^{-1} \dot{\bar{J}})^{-1} J\dot{q}_t + hJM^{-1} \dot{\tau}. \tag{7}
\]

In our experiment a solution can always be found at each time step with 0.5% convergence tolerance. The number of evaluations of \( G \) is often fewer than 10.
D. Implementation

A contact patch $P$ in the real world will always be a 2D surface. However, when $P$ degenerates to nearly an edge or a point, in practice, the dimension of the controllable space of the friction impulse will reduce. Since the dimension of $P$ is available from the collision detector $D$, we utilize this information to improve learning accuracy by treating three types of $P$ separately: a surface (2D), a line (1D), or a point (0D). For each type, we train a specific classifier and a regressor. Using separate neural networks allows the regressors to have different output dimensions according to the dimension of controllable space of the friction impulse.

The same set of training data can be used to train the classifiers and the regressors. The data collection involves throwing objects to each other with different initial velocities. Since $\tau$ is not part of the input of the learned models, we do not need to apply various $\tau$ during training sample generation, greatly simplifying the data collection process. We record the entire trajectory for each throw and extract the state of every contact instance: $q_t, \dot{q}_t$. By evaluating $\dot{q}_t$ at the patch, we can identify and label static and dynamic cases. To determine detach cases and to calculate the training output for the regressors, we need to recover the contact impulse $p$ for each contact instance: $p = M(q_{t+1} - q_t) - hg$, where $g$ is the gravitational force. If $p$ is near zero, we label this contact instance "detach".

The range of initial velocities is chosen to cover the range of the anticipated collision impulses during testing. We found that the choice of input representation significantly affects the accuracy of the learned models. Our experiments show that representing the 3D orientation of the rigid body as a rotation matrix outperforms other representations of $SO(3)$. We also found that including two redundant features—the position of the center of $P$ in the body frame and the velocity at the center of $P$—reduces the errors of the regressors.

IV. Evaluation

We evaluated our data-augmented contact model on rigid bodies with different 3D shapes, a 3-linked articulated rigid body chain, and an object with anisotropic friction coefficient. To demonstrate the complexity of 3D collision, we also included one 2D example for comparison. The data were collected in a simulated environment using a generic physics engine DART [31], which approximates the Coulomb friction cone using a square pyramid with a single friction coefficient.

Although our method is agnostic to the representation of classifiers or regressors, we used feed-forward neural networks for their expressiveness as function approximators. A standard cross-entropy error or MSE was used for the loss function. We used the same collision data to train classifiers and regressors. The hyper-parameters are shown in Fig. 3.

We evaluated our results against two baselines. The first baseline serves as the ground truth (GT), which computes contact impulses from the same contact handling routine (i.e. LCP solver in DART) used to collect the training data. The second baseline is a purely data-driven (PDD) approach which learns regressors to directly predict impulse from the rigid body state, without first predicting the contact state using classifiers. The input representation, network architecture and learning algorithm are the same between PDD and our regressors. However, we gave PDD 50% more training data to reach comparable accuracy (Fig. [3]).

A. Accuracy of prediction

We generated 40,000 individual collisions to test each learned classifier and regressor. The accuracies are shown in Fig. 3. Note that PDD can reach 90%+ accuracy when predicting individual collisions.

B. Single rigid body

We threw a box and a pentagon prism to the ground under gravity from various initial positions and velocities. The range of initial states and the geometries of the rigid bodies are detailed in Fig. 4. Each simulated trajectory contains 800 time steps, equivalent to 1.6 second of motion. For each rigid body, we simulated 100 trajectories with random initial states and reported three metrics: the average errors of the final horizontal distance, of the final orientation, and of the first collision impulse of each trajectory.

Fig. 5 shows the results in comparison with the two baselines. In most cases, our method matches GT closely and is significantly better than PDD, demonstrating the advantages of using a classifier and analytical solutions. It is worth noting that both our method and PDD achieve similar accuracy in learning the regressors, but our method has much lower error in predicting the impulses. This is because when a collision instance is classified as "static" or "detach", our method solves for an analytical solution which adds no error to the simulation. We also notice that PDD produces large

| Hyper-parameters and testing accuracy. | | | | | |
|---|---|---|---|---|---|
| Hidden layers | Epochs | Dropout | Training samples | Accuracy point | Accuracy line | Accuracy surface |
| Our 3D box classifier | 256-100-80 | 100 | 10% | 150K | 98% | 97% | 98% |
| Our 3D box regressor | 512-360-180 | 100 | 5% | 252K | 97% | 97% | 92% |
| PDD 3D box regressor | 512-360-180 | 100 | 5% | 225K | 96% | 91% | 95% |
| Our 3D Pentagon classifier | 256-100-80 | 100 | 10% | 150K | 98% | 97% | 98% |
| Our 3D pentagon regressor | 640-480-240 | 100 | 5% | 225K | 99% | 94% | 91% |
| PDD 3D pentagon regressor | 640-480-240 | 100 | 5% | 225K | 99% | 94% | 91% |

2The later collisions cannot be compared to the ground truth because the motions start to deviate after the first collision.

Fig. 3. Range of initial states and the rigid bodies used in our experiments. $h_0$, $R$, $\omega$, $v$ indicate the initial height, 3D orientation, angular velocity, and linear velocity respectively.
errors in distance and orientation. This is because the small but persistent errors in impulse often result in perpetual movements instead of letting the rigid body come to rest. The erroneous behavior further highlights the advantage of identifying static cases and enforcing analytical constraints for those cases. For comparison, we also tested PDD and our method on a 2D rectangle. PDD performs much better on the 2D problem and is comparable to our method. This seems to suggest that PDD can only simulate consecutive bounces well when its regressor has a high accuracy ($\approx 98\%$), which is much harder to achieve on 3D problems.

Since 3D motions involve much more complex contact behaviors than 2D motions, we also compared the sequence of contact events in addition to the final state of the trajectory. Fig. 6 shows the contact event sequences of the 3D box from five random throws. To represent a sequence, we used integers (1-8) to label the contacting vertices, lower case letters (a to l) to label the contacting edges, and upper case letters (A to F) to label contacting faces. The results show that our method produces similar contact events to GT while PDD produces wildly different contact events.

C. Articulated rigid bodies

We demonstrated our method on an articulated three link system connected by two revolute joints. The top of the first link is pinned to a fixed world space location. The chain started at a horizontal position and swung passively to the ground under gravity. We compared our method to ground truth and showed the motion sequences in the supplementary video. Though our method is only trained on the contact instances between an isolated distal body (i.e. the third link) and the ground, we show that both contact impulses and joint constraint forces can be predicted or solved accurately.

D. Anisotropic friction cone

We created a fictitious material which has an anisotropic friction cone. The friction coefficient of the ground is 1.5 along z-axis and 0.75 along x-axis. We collected training data from this simulated scenario and learned the classifiers and regressors using the same algorithms. During testing, we threw a box to the ground in eight directions. For each direction, we oriented the initial orientation and velocity to align with the throwing direction. Fig. 7 shows the distances traveled for each direction using our contact model (top) and using the GT simulator (bottom). The results show that our method is able to predict the outcome of collision for anisotropic materials.

V. CONCLUSION AND LIMITATIONS

We introduced a data-augmented contact model that predicts the contact behaviors for a particular pair of near-rigid bodies or articulated rigid body systems. We evaluated our method on a set of 3D examples using simulated training data. The promising results indicate that our work could be a first step toward a contact model capable of predicting the 3D contact behaviors of real-world, near-rigid materials.

Our method assumes that the applied forces can be integrated prior to solving the constraint force. This assumption is not true for some special cases. Consider, in Case A, a 1kg object with zero initial velocity on a surface is under an applied force $f$ horizontal to the surface, and in Case B, without an external force, the same object has an initial velocity $h$. If the magnitude of $f$ happens to be $\mu g < ||f|| < \mu_s g$, where $\mu$ and $\mu_s$ are the dynamic and static friction coefficients and $g$ is the magnitude of gravitational force applied on the object, Case A will have zero velocity while Case B will have a velocity of $h||f|| - h\mu_d g$ at the end of the time step. One possible way to address this issue is to include $\tau$ as part of the input vector to the neural networks, instead of integrating it into $q_t$. We can also represent the applied force as a distribution of forces instead of a single aggregated force vector. This representation might produce more accurate prediction for small-scale contact impulses.

When collecting the training data from the real world, an isolated distal part of the robot will be used to create the collisions with the surface. The range of collision impulses should cover that of the anticipated collisions during the operation of the full robot. Although the data collection can be conducted in isolation without involving the entire robot, the data-efficiency remains a major concern. Since the dimension of the input and output space is relatively low, it is possible to use other function approximators, such as support vector machines or Gaussian processes, which might be more data-efficient than neural networks.

Finally, our current algorithm assumes that one of the objects is stationary, and does not handle simultaneous contacts of multiple bodies, which limits its usage in many manipulation tasks. As immediate future directions, we plan to extend Algorithm 1 to two moving bodies by including states of both objects as input.
REFERENCES

[1] J. C. Zagal, J. Ruiz-del Solar, and P. Vallejos, “Back to reality: Crossing the reality gap in evolutionary robotics,” IFAC Proceedings Volumes, vol. 37, no. 8, pp. 834–839, 2004.

[2] B. Brogliato and B. Brogliato, Nonsmooth mechanics. Springer, 1999.

[3] C. Lemke and J. Howson, “Equilibrium points of bimatrix games,” SIAM Journal on Applied Mathematics, vol. 12, 1964.

[4] D. Stewart and J. C. Trinkle, “An implicit time-stepping scheme,” Computer Methods in Applied Mechanics and Engineering, vol. 155, 1998.

[5] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, “More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 30–37, IEEE, 2016.

[6] E. Popova and V. L. Popov, “The research works of Coulomb and Amontons and generalized laws of friction,” Friction, vol. 3, 2015.

[7] J. Oden and J. Martins, “Models and computational methods for dynamic friction phenomena,” Computer methods in applied mechanics and engineering, vol. 52, 1985.

[8] H. Olsson, K. J. Åström, C. C. De Wit, M. Gäfvert, and P. Lischinsky, “Friction models and friction compensation,” Eur. J. Control, vol. 4, no. 3, pp. 176–195, 1998.

[9] S. Goyal, A. Ruina, and J. Papadopoulos, “Limit surface and moment function descriptions of planar sliding,” in Robotics and Automation (ICRA), 1989 IEEE International Conference on, pp. 794–799, IEEE, 1989.

[10] D. Baraff, “Analytical methods for dynamic simulation of non-penetrating rigid bodies,” in ACM SIGGRAPH Computer Graphics, vol. 23, pp. 223–232, ACM, 1989.

[11] D. Stewart and J. C. Trinkle, “A gauss-seidel like algorithm to solve frictional contact problems,” Computer Methods in Applied Mechanics and Engineering, vol. 155, 1998.

[12] M. Anitescu and F. A. Potra, “A time-stepping method for stiff multibody dynamics with contact and friction,” International Journal for Numerical Methods in Engineering, vol. 55, 2002.

[13] E. Todorov, “Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo,” Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 6054–6061, 2014.

[14] B. V. Mirtich, Impulse-based Dynamic Simulation of Rigid Body Systems. PhD thesis, University of California at Berkeley, 1996.

[15] R. Grzeszczuk, D. Terzopoulos, and G. Hinton, “Neuroanimatror: Fast neural network emulation and control of physics-based models,” in Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 9–20, ACM, 1998.

[16] S.-W. Hsu and J. Keyser, “Statistical simulation of rigid bodies,” in Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 139–148, ACM, 2009.

[17] S. Jeong, B. Solenthaler, M. Pollefeys, M. Gross, et al., “Data-driven fluid simulations using regression forests,” ACM Transactions on Graphics (TOG), vol. 34, no. 6, p. 199, 2015.

[18] C. Yang, X. Yang, and X. Xiao, “Data-driven projection method in fluid simulation,” Computer Animation and Virtual Worlds, no. 3–4, pp. 415–424, 2016.

[19] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, “Accelerating eulerian fluid simulation with convolutional networks,” in International Conference on Machine Learning, pp. 3424–3433, 2017.

[20] M. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, “A compositional object-based approach to learning physical dynamics,” in Proceedings of the 5th International Conference on Learning Representations, 2016.

[21] A. Byravan and D. Fox, “Sc3-nets: Learning rigid body motion using deep neural networks,” in Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 173–180, IEEE, 2017.

[22] A. Lerer, S. Gross, and R. Fergus, “Learning physical intuition of block towers by example,” in International Conference on Machine Learning, pp. 430–438, 2016.

[23] J. Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason, “A convex polynomial force-motion model for planar sliding: Identification and application,” in Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 372–377, IEEE, 2016.

[24] M. Bauza and A. Rodriguez, “A probabilistic data-driven model for planar pushing,” in Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 3008–3015, IEEE, 2017.

[25] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez, “Learning data-efficient rigid-body contact models: Case study of planar impact,” in Conference on Robot Learning, pp. 388–397, 2017.

[26] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez, “Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing,” arXiv preprint arXiv:1808.03246, 2018.

[27] A. Kloss, S. Schaal, and J. Bohg, “Combining learned and analytical models for predicting action effects,” arXiv:1710.04102, 2018.

[28] D. Baraff, Dynamic Simulation of Non-penetrating Rigid Bodies. PhD thesis, Cornell University, 1992.

[29] R. W. Cottle and G. B. Dantzig, “Complementary pivot theory of mathematical programming,” Linear Algebra and its Applications, vol. 1, no. 1, pp. 103 – 125, 1968.

[30] M. J. Powell, “A hybrid method for nonlinear equations,” Numerical methods for nonlinear algebraic equations, 1970.

[31] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, “DART: Dynamic animation and robotics toolkit,” The Journal of Open Source Software, 2018.