Data-Augmented Contact Model for Rigid Body Simulation

Yifeng Jiang YIFENGJ@STANFORD.EDU
Stanford University, California, United States

Jiazheng Sun JIAZHENGSUN96@GMAIL.COM
Amazon.com, Inc., Washington, United States

C. Karen Liu KARENLIU@CS.STANFORD.EDU
Stanford University, California, United States

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

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, maintaining same simulation speed as the number of contact points increases for detailed geometries. Supplemental video: `https://shorturl.at/eilwX`

Keywords: Physics Simulation Algorithms, Dynamics Learning, Contact Modeling

1. Introduction

The use of virtual simulations has been crucial to the recent progress in building learning-based robot controllers. With recent works showing that improving the rendering realism of simulated perception Rao et al. (2020) or the accuracy of robot motor dynamics Hwangbo et al. (2019) can improve the performance of learned controllers, more accurate modeling of the contact dynamics would also help the robot better understand and interact with the environment it resides. Most recent robot controllers are still developed from off-the-shelf physics simulators which use the idealized Coulomb friction model, an empirical construct to approximate the changes between two physical regimes (static friction vs dynamic friction). However, the Coulomb model assumes linear relationship between normal force and frictional force and uses a single friction coefficient to represent an isometric friction cone. The computation of contact force also involves approximation and arbitrary decisions. For example, many existing simulators formulate a Linear Complementarity Program (LCP), which solutions are not unique except for the frictionless case Brogliato and Brogliato (1999). Depending on the initial guesses and numerical methods used for solving the LCP problem (e.g. Lemke method Lemke and Howson (1964) vs Gauss-Seidel algorithm Jourdan et al.

(1998)), the resulting contact forces can be drastically different. These existing issues suggest that a computational contact model grounded by observed data can be a desired alternative.

While recent work has shown that physical phenomena can be learned from data and approximated by neural networks, 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 combining analytical solutions and empirical data collected for a particular scenario (e.g. a specific robot foot colliding a specific surface), such that the simulated trajectories better match the observed data.

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 observed data to model and validate the less-understood static friction boundary. 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 simulation speed as the number of contact points increases for real-world objects with detailed geometries.

Our algorithm also applies to articulated systems, distinguishing our framework from most data-driven contact models. 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 learning the contact can be collected using replicas of the disassembled end-effector, without putting the entire robotic system at risk.

As our work focuses on a full 3D scenario in which the object can bounce, slide, or spin in all directions, existing contact data such as MIT Push Yu et al. (2016) are not diverse enough for our evaluation. Instead, we learn the data-driven models from synthesised collision data in simulations. Evaluations show that our data-augmented model matches ground-truth contact behaviors of both single and articulated rigid body systems, and better than a purely statistical model.

2. 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, inadequacies and controversies of Coulomb's law have been extensively studied Popova and Popov (2015). For example, Oden and Martins (1985) 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 Olsson et al. (1998). There is no one model that is more accurate than the others in all scenarios. Goyal et al. (1989) proposed the concept of limit surface to enclose all possible friction forces on an object during planer sliding. Our 3D model also works on the rigid-body instead of the point-contact level, and loosens 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 Baraff (1989); Stewart and Trinkle (2000); Anitescu and Potra (2002). In contrast, Todorov (2014) relaxed contact constraints to solve a convex optimization problem at each time step.

With the emergence of simulation-based controller learning, developing data-driven physics simulators capable of predicting the real world has been actively studied. Chang et al. (2016); Byravan and Fox (2017); Lerer et al. (2016) trained neural networks to learn physical intuitions from vision perception, Zhou et al. (2016) built a data-efficient model for the limit surface of an object during planar sliding, Bauza and Rodriguez (2017) built a probabilistic model for planer sliding that takes into account the stochasticity of frictional forces, and Fazeli et al. (2017) 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. Our work is instead a data-augmented one, incorporating observed data while preserving fundamental contact constraints analytically. Contact detection (not handling) could also be replaced with learnable components via learning contact Jacobian Pfrommer et al. (2020) or differentiable shape parameterization Strecke and Stueckler (2021). System identification Chebotar et al. (2019) serves as another approach to improve simulation fidelity without requiring new simulation models, with recent works demonstrating promising results by utilizing gradient-free reinforcement learning Jeong et al. (2019); Jiang et al. (2021), or making simulators differentiable Heiden et al. (2021); Le Lidec et al. (2021); Degrave et al. (2019); Jatavallabhula et al. (2021).

The advantages of integrating neural networks with analytical models have been demonstrated recently. Ajay et al. (2018) introduced a method that trains a recurrent neural network to predict the deviation between real-world and simulated contact trajectories, extended in Fazeli et al. (2020) for better long-term accuracy. Kloss et al. (2018) proposed to combine a neural network for perception with a physics model for prediction, and Pizzuto and Mistry (2021) added a non-penetration loss to augment the data loss. Instead of treating the physics engine as a black box and correcting its output, our work directly improves the full 3D contact handling using learned function approximators. We also show that our learned contact models can be reused in new articulated systems without retraining.

3. 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).

3.1. 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 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 in contact, denoted as $\mathcal{P}$. We assume that the normal direction of the contact patch is well-defined.

3.2. Single rigid body

Algorithm 1: Single body contact model: H

```
Input  : $\mathbf{q}_t, \dot{\mathbf{q}}_t, \boldsymbol{\tau}, \mathcal{P}$
Output: $\mathbf{p}$
1  $\dot{\mathbf{q}}^{(1)} = \dot{\boldsymbol{q}}_t + h\mathbf{M}^{-1}\boldsymbol{\tau}$;
2  $c \leftarrow C(\mathbf{q}_t, \dot{\mathbf{q}}^{(1)})$;
3  if $c ==$ "static" then
4      AddPositionConstraint($\mathcal{P}$);
5      $\mathbf{p} \leftarrow$ SolveConstraint($\mathbf{q}_t, \dot{\mathbf{q}}^{(1)}$);
6      RemovePositionConstraint($\mathcal{P}$);
7  else if $c ==$ "dynamic" then
8      $\mathbf{p}_f \leftarrow R(\mathbf{q}_t, \dot{\mathbf{q}}^{(1)})$;
9      $\dot{\mathbf{q}}^{(2)} = \dot{\mathbf{q}}^{(1)} + \mathbf{M}^{-1}\mathbf{T}_f\mathbf{p}_f$;
10     $\mathbf{p}_n \leftarrow$ 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     $\mathbf{p} = \mathbf{0}$;
14 end
15 return $\mathbf{p}$
```

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 can be expressed as a function $\mathbf{p} = H(\mathbf{q}_t, \dot{\mathbf{q}}_t, \boldsymbol{\tau}, \mathcal{P})$, which maps the pre-contact state (i.e. $\mathbf{q}_t$ and $\dot{\mathbf{q}}_t$), applied forces $\boldsymbol{\tau}$, 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

$$\dot{\mathbf{q}}_{t+1} = \dot{\mathbf{q}}_t + \mathbf{M}(\mathbf{q}_t)^{-1}(h\boldsymbol{\tau} + \mathbf{p}), \tag{1}$$

$$\mathbf{q}_{t+1} = \text{Integrate}(\mathbf{q}_t, \dot{\mathbf{q}}_{t+1}), \tag{2}$$

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 $\dot{\mathbf{q}}_t$ to an intermediate velocity by explicitly integrating the applied force $\boldsymbol{\tau}$: $\dot{\mathbf{q}}^{(1)} = \dot{\mathbf{q}}_t + h\mathbf{M}^{-1}\boldsymbol{\tau}$. 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 $C(\mathbf{q}_t, \dot{\mathbf{q}}^{(1)})$ 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).

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 $\mathbf{v}_p = \mathbf{0}$, 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 $\mathbf{v}_p(\mathbf{p}) = \mathbf{0}$ has a unique solution. Therefore, if a state is classified as "static" by a highly accurate classifier C, the solution of $\mathbf{v}_p(\mathbf{p}) = \mathbf{0}$ will satisfy Properties (1)-(3).

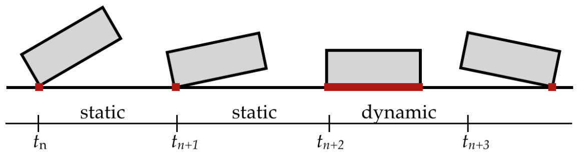

Figure 1: Illustration of "static" and "dynamic" cases. The classifier predicts "static" at t_n and t_{n+1}. We analytically solve a contact impulse to ensure the contact patch (shown in red) has zero velocity at the end of t_n and t_{n+1}. The classifier predicts "dynamic" at t_{n+2} and expects the contact patch to change at the end of t_{n+2}.

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 $\mathbf{p}_f = (p_x, p_z, m_y)^T$ and a normal impulse $\mathbf{p}_n = (p_y, \tilde{m}_x, \tilde{m}_z)^T$. 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 $\mathbf{p}_n$ such that Properties (1)-(3) are satisfied [1]. Therefore, we train a regressor to only predict $\mathbf{p}_f$ and analytically calculate the unique solution for $\mathbf{p}_n$ based on the predicted $\mathbf{p}_f$. Specifically, Algorithm 1 in the dynamic case first predicts $\mathbf{p}_f$ using the trained regressor $R(\mathbf{q}_t, \dot{\mathbf{q}}^{(1)})$ (Line 8) and then integrates $\mathbf{p}_f$ to obtain

1. The actual distribution of normal force over the contact patch is still undetermined. Baraff (1992)

a second intermediate velocity: $\dot{\mathbf{q}}^{(2)} = \dot{\mathbf{q}}^{(1)} + \mathbf{M}^{-1}\mathbf{T}_f\mathbf{p}_f$ (Line 9). Here $\mathbf{T}_f \in \mathbb{R}^{6\times 3}$ transforms $\mathbf{p}_f$ to the generalized coordinates. The unique solution for $\mathbf{p}_n$ can be solved by any routine that respects normal complementaries. Our algorithm uses a Danzig-like positive definite LCP solver for frictionless contacts Cottle and Dantzig (1968") (Line 10). Finally, we combine the analytical $\mathbf{p}_n$ and predicted $\mathbf{p}_f$ to obtain $\mathbf{p}$ in generalized coordinates (Line 11).

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

Detach case In the detach case, $\mathcal{P}$ is predicted to have positive normal velocity and leave the surface at the next time step. We thus set $\mathbf{p} = \mathbf{0}$, 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.

3.3. Articulated rigid bodies

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

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

$$\dot{\mathbf{q}}_{t+1} = \dot{\mathbf{q}}_t + h\mathbf{M}^{-1}\boldsymbol{\tau} + h\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_j + \mathbf{M}^{-1}\mathbf{p}, \quad (3)$$

where $\mathbf{M}(\mathbf{q}_t)$ is the generalized mass matrix for the distal body and $\mathbf{J}(\mathbf{q}_t)$ 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

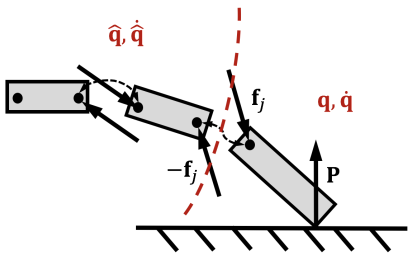
$\hat{\mathbf{q}}, \dot{\hat{\mathbf{q}}}$
$\mathbf{f}_j$
$\mathbf{q}, \dot{\mathbf{q}}$
$-\mathbf{f}_j$
$\mathbf{p}$

Figure 2: An articulated rigid body system which consists of a distal body whose state is $(\mathbf{q}, \dot{\mathbf{q}})$ and other bodies in the upstream system $(\hat{\mathbf{q}}, \dot{\hat{\mathbf{q}}})$.

$$\mathbf{v}_{t+1} = \mathbf{J}\dot{\mathbf{q}}_t + h\mathbf{J}\mathbf{M}^{-1}\boldsymbol{\tau} + h\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_j + \mathbf{J}\mathbf{M}^{-1}\mathbf{p}. \quad (4)$$

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

$$\hat{\mathbf{v}}_{t+1} = \hat{\mathbf{J}}\dot{\hat{\mathbf{q}}}_t + h\hat{\mathbf{J}}\hat{\mathbf{M}}^{-1}\hat{\boldsymbol{\tau}} - h\hat{\mathbf{J}}\hat{\mathbf{M}}^{-1}\hat{\mathbf{J}}^T\mathbf{f}_j, \quad (5)$$

where $\hat{\mathbf{M}}$ and $\hat{\mathbf{J}}$ 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 $\mathbf{f}_j$ such that $\mathbf{v}_{t+1} - \hat{\mathbf{v}}_{t+1} = \mathbf{0}$:

$$G(\mathbf{f}_j) = \mathbf{v}_{t+1} - \hat{\mathbf{v}}_{t+1} = \mathbf{A}\mathbf{f}_j + \mathbf{J}\mathbf{M}^{-1}H(\mathbf{f}_j) + \mathbf{c} = \mathbf{0}, \quad (6)$$

where $\mathbf{A} = h\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T + h\hat{\mathbf{J}}\hat{\mathbf{M}}^{-1}\hat{\mathbf{J}}^T$ and $\mathbf{c} = \mathbf{J}\dot{\mathbf{q}}_t + h\mathbf{J}\mathbf{M}^{-1}\boldsymbol{\tau} - (\hat{\mathbf{J}}\dot{\hat{\mathbf{q}}}_t + h\hat{\mathbf{J}}\hat{\mathbf{M}}^{-1}\hat{\boldsymbol{\tau}})$ are constants in the equation given $\mathbf{q}_t$ and $\dot{\mathbf{q}}_t$. $H(\mathbf{f}_j)$ is a shorthand for $H(\mathbf{q}_t, \dot{\mathbf{q}}_t, \boldsymbol{\tau} + \mathbf{J}^T\mathbf{f}_j, \mathcal{P})$, which outputs $\mathbf{p}$ depending on $\mathbf{f}_j$.

Algorithm 2: Contact solver for articulated rigid-body chain

```
Input  : $(\mathbf{q}_t, \dot{\mathbf{q}}_t, \boldsymbol{\tau}), (\hat{\mathbf{q}}_t, \dot{\hat{\mathbf{q}}}_t, \hat{\boldsymbol{\tau}}), \mathcal{P}$
Output: $\mathbf{q}_{t+1}, \dot{\mathbf{q}}_{t+1}, \hat{\mathbf{q}}_{t+1}, \dot{\hat{\mathbf{q}}}_{t+1}$
1  Initialize $\mathbf{f}_j$ using Eq. 7;
2  Calculate $\mathbf{J}, \mathbf{M}, \hat{\mathbf{J}}, \hat{\mathbf{M}}$;
3  while solver not terminated do
4      Evaluate $G(\mathbf{f}_j)$ using Eq. 6;
5      Update $\mathbf{f}_j$ according to Powell's method;
6  end
7  $\mathbf{p} \leftarrow H(\mathbf{q}_t, \dot{\mathbf{q}}_t, \boldsymbol{\tau} + \mathbf{J}^T\mathbf{f}_j, \mathcal{P})$;
8  $\dot{\mathbf{q}}_{t+1} = \dot{\mathbf{q}}_t + h\mathbf{M}^{-1}\boldsymbol{\tau} + h\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_j + \mathbf{M}^{-1}\mathbf{p}$;
9  $\mathbf{q}_{t+1}$ = Integrate($\mathbf{q}_t, \dot{\mathbf{q}}_{t+1}$);
10 $\dot{\hat{\mathbf{q}}}_{t+1} = \dot{\hat{\mathbf{q}}}_t + h\hat{\mathbf{M}}^{-1}\hat{\boldsymbol{\tau}} - h\hat{\mathbf{M}}^{-1}\hat{\mathbf{J}}^T\mathbf{f}_j$;
11 $\hat{\mathbf{q}}_{t+1}$ = Integrate($\hat{\mathbf{q}}_t, \dot{\hat{\mathbf{q}}}_{t+1}$);
12 return $\mathbf{q}_{t+1}, \dot{\mathbf{q}}_{t+1}, \hat{\mathbf{q}}_{t+1}, \dot{\hat{\mathbf{q}}}_{t+1}$
```

We solve Eq. 6 using Powell hybrid method Powell (1970), 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 $G(\mathbf{f}_j)$ and an initial guess. Using the heuristic that assumes $\hat{\mathbf{v}}_{t+1} = \mathbf{0}$, we compute the initial $\mathbf{f}_j$ by

$$\mathbf{f}_j = (h\hat{\mathbf{J}}\hat{\mathbf{M}}^{-1}\hat{\mathbf{J}}^T)^{-1}\hat{\mathbf{J}}(\dot{\hat{\mathbf{q}}}_t + h\hat{\mathbf{M}}^{-1}\hat{\boldsymbol{\tau}}). \qquad (7)$$

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.

3.4. Implementation

A contact patch $\mathcal{P}$ in the real world will always be a 2D surface. However, when $\mathcal{P}$ 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 $\mathcal{P}$ is available from the collision detector D, we utilize this information to improve learning accuracy by treating three types of $\mathcal{P}$ 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 $\boldsymbol{\tau}$ is not part of the input of the learned models, we do not need to apply various $\boldsymbol{\tau}$ 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: $\mathbf{q}_t, \dot{\mathbf{q}}_t$. By evaluating $\dot{\mathbf{q}}_t$ 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 $\mathbf{p}$ for each contact instance: $\mathbf{p} = \mathbf{M}(\dot{\mathbf{q}}_{t+1} - \dot{\mathbf{q}}_t) - h\mathbf{g}$, where $\mathbf{g}$ is the gravitational force. If $\mathbf{p}$ 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 rotation 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 $\mathcal{P}$ in the body frame and the velocity at the center of $\mathcal{P}$—reduces the errors of the regressors.

4. 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 ground-truth data were collected in a simulated environment using a generic physics engine DART Lee et al. (2018).

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

	Hidden layers	Epochs	Dropout	Training samples	Accuracy point	Accuracy line	Accuracy surface
Our 3D box classifier	256-180-80	100	10%	150K	98%	97%	98%
Our 3D box regressor	512-360-180	1000	5%		97%	91%	92%
PDD 3D box regressor	512-360-180	1000	5%	225K	96%	91%	95%
Our 3D pentagon classifier	256-180-80	100	10%	150K	98%	97%	98%
Our 3D pentagon regressor	640-480-240	1000	5%		96%	84%	91%
PDD 3D pentagon regressor	640-480-240	1000	5%	225K	98%	91%	94%

Figure 3: Hyper-parameters and testing accuracy.

We evaluated our results against two baselines. The first baseline is the ground truth (GT), i.e. contact handling of DART. The second baseline is a purely data-driven (PDD) approach which learns regressors to directly predict impulse from the rigid body state, without using classifiers or solving analytical constraints. 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 one-step accuracy (Fig. 3).

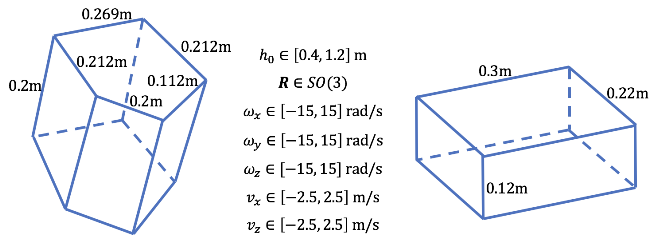
0.269m
0.212m
0.212m
0.2m
0.112m
0.2m
$h_0 \in [0.4, 1.2]$ m
$\boldsymbol{R} \in SO(3)$
$\omega_x \in [-15, 15]$ rad/s
$\omega_y \in [-15, 15]$ rad/s
$\omega_z \in [-15, 15]$ rad/s
$v_x \in [-2.5, 2.5]$ m/s
$v_z \in [-2.5, 2.5]$ m/s
0.3m
0.22m
0.12m

Figure 4: Range of initial states and the rigid bodies used in our experiments.

4.1. Single step 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.

4.2. Predicting trajectories of 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, where h_0, $\mathbf{R}$, ω, v indicate the ranges of the initial height, 3D orientation, angular velocity, and linear velocity respectively. 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

	Box 3D			Pentagon Prism 3D			Box 2D		
	GT Mean	PDD Err. Mean	Our Err. Mean	GT Mean	PDD Err. Mean	Our Err. Mean	GT Mean	PDD Err. Mean	Our Err. Mean
Final Position	0.841	0.531	0.039	0.887	0.434	0.053	0.926	0.058	0.088
Final Orientation	2.265	2.258	0.243	2.335	2.114	0.355	1.665	0.285	0.304
First Impulse	2.865	0.263	0.033	3.167	0.207	0.014	2.755	0.087	0.009

Figure 5: Results of the single body experiments.

reported three metrics: the average errors of the final horizontal distance, of the final orientation, and of the first collision impulse [2] of each trajectory.

Fig. 5 shows the results in comparison with the two baselines, where for each metric, we show the mean of GT, the average error of PDD compared to GT, and the average error of our method compared to GT. 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 correctly 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 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 planar throwing problem and is comparable to our method. This seems to suggest that PDD can only simulate consecutive bounces well when its regressor has a very high accuracy ($\approx 98\%$), which is much harder to achieve in 3D problems.

Since 3D motions involve much more complex contact behaviors than 2D, we also compared the sequence of contact events (rolling, spinning etc.) 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.

	Trial #1	Trial #2	Trial #3	Trial #4	Trial #5
GT	$7j8l3c4eF$	$8h54c3lF$	$5h8lF$	$1g6gDeDgD$	$1a2kEgE$
Ours	$7j834eF$	$8h54c3lF$	$5h8F$	$16f5DgDeDgDeD$	$1a27kE$
PDD	$7j84c383$ $c4d1g678$	$8h5e43$ $2a1627$	$5h3c4e5h45$ $4c3l8h5e13$ $5Fl3434534$	$1g672b34216$	$1a287i61$ $d27j8l3$

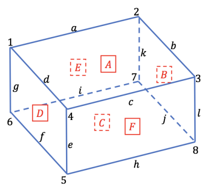

Figure 6: Contact event sequences of five random trials.

4.3. Predicting trajectories of 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 swang passively to the ground under gravity. We compared our method to ground truth

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

	0°	45°	90°	135°	180°	225°	270°	315°
Our Distance	0.905	0.754	0.721	0.748	1.154	0.778	0.695	0.748
GT Distance	0.929	0.775	0.752	0.775	0.929	0.775	0.691	0.775

Figure 7: A box thrown in eight directions onto a surface with anisotropic materials.

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.

4.4. 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.

5. 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.

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.

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. Finally, combining our method with learnable contact detectors could further increase expressiveness of the model.

References

Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. *arXiv preprint arXiv:1808.03246*, 2018.

Mihai Anitescu and Florian A Potra. A time-stepping method for stiff multibody dynamics with contact and friction. *International Journal for Numerical Methods in Engineering*, 55, 2002.

David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *ACM SIGGRAPH Computer Graphics*, volume 23, pages 223–232. ACM, 1989.

David Baraff. *Dynamic Simulation of Non-penetrating Rigid Bodies*. PhD thesis, Cornell University, 1992.

Maria Bauza and Alberto Rodriguez. A probabilistic data-driven model for planar pushing. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3008–3015. IEEE, 2017.

Bernard Brogliato and B Brogliato. *Nonsmooth mechanics*. Springer, 1999.

Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 173–180. IEEE, 2017.

Michael Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *Proceedings of the 5th International Conference on Learning Representations*, 2016.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

Richard W. Cottle and George B. Dantzig. Complementary pivot theory of mathematical programming. *Linear Algebra and its Applications*, 1(1):103 – 125, 1968”. ISSN 0024-3795. doi: https://doi.org/10.1016/0024-3795(68)90052-9. URL http://www.sciencedirect.com/science/article/pii/0024379568900529.

Jonas Degrave, Michiel Hermans, Joni Dambre, and Francis wyffels. A differentiable physics engine for deep learning in robotics. *Frontiers in Neurorobotics*, 13:6, 2019. ISSN 1662-5218. doi: 10.3389/fnbot.2019.00006. URL https://www.frontiersin.org/article/10.3389/fnbot.2019.00006.

Nima Fazeli, Samuel Zapolsky, Evan Drumwright, and Alberto Rodriguez. Learning data-efficient rigid-body contact models: Case study of planar impact. In *Conference on Robot Learning*, pages 388–397, 2017.

Nima Fazeli, Anurag Ajay, and Alberto Rodriguez. Long-horizon prediction and uncertainty propagation with residual point contact learners. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7898–7904. IEEE, 2020.

Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Limit surface and moment function descriptions of planar sliding. In *Robotics and Automation (ICRA), 1989 IEEE International Conference on*, pages 794–799. IEEE, 1989.

Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481, 2021. doi: 10.1109/ICRA48506.2021.9560935.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=c_E8kFWfhp0.

Rae Jeong, Jackie Kay, Francesco Romano, Thomas Lampe, Tom Rothorl, Abbas Abdolmaleki, Tom Erez, Yuval Tassa, and Francesco Nori. Modelling generalized forces with reinforcement learning for sim-to-real transfer. *arXiv preprint arXiv:1910.09471*, 2019.

Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C. Karen Liu, Sergey Levine, and Jie Tan. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2884–2890, 2021. doi: 10.1109/ICRA48506.2021.9561731.

F. Jourdan, P. Alart, and M. Jean. A gauss-seidel like algorithm to solve frictional contact problems. *Computer Methods in Applied Mechanics and Engineering*, 155, 1998.

Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects. *arXiv:1710.04102*, 2018.

Quentin Le Lidec, Igor Kalevatykh, Ivan Laptev, Cordelia Schmid, and Justin Carpentier. Differentiable simulation for physical system identification. *IEEE Robotics and Automation Letters*, 6 (2):3413–3420, 2021. doi: 10.1109/LRA.2021.3062323.

Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 2018. doi: 10.21105/joss.00500. URL https://doi.org/10.21105/joss.00500.

C. Lemke and J. Howson. Equilibrium points of bimatrix games. *SIAM Journal on Applied Mathematics*, 12, 1964.

Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *International Conference on Machine Learning*, pages 430–438, 2016.

JT Oden and JAC Martins. Models and computational methods for dynamic friction phenomena. *Computer methods in applied mechanics and engineering*, 52, 1985.

Henrik Olsson, Karl Johan Åström, Carlos Canudas De Wit, Magnus Gäfvert, and Pablo Lischinsky. Friction models and friction compensation. *Eur. J. Control*, 4(3):176–195, 1998.

Samuel Pfrommer, Mathew Halm, and Michael Posa. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. *arXiv preprint arXiv:2009.11193*, 2020.

Gabriella Pizzuto and Michael Mistry. Physics-penalised regularisation for learning dynamics models with contact. In *Learning for Dynamics and Control*, pages 611–622. PMLR, 2021.

Elena Popova and Valentin L. Popov. The research works of Coulomb and Amontons and generalized laws of friction. *Friction*, 3, 2015.

Michael JD Powell. A hybrid method for nonlinear equations. *Numerical methods for nonlinear algebraic equations*, 1970.

Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

David Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. In *Robotics and Automation (ICRA), 2000 IEEE International Conference on*, volume 1, pages 162–169. IEEE, 2000.

Michael Strecke and Joerg Stueckler. DiffSDFSim: Differentiable rigid-body dynamics with implicit shapes. In *International Conference on 3D Vision (3DV)*, December 2021.

Emanuel Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6054–6061, 2014.

Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto 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*, pages 30–37. IEEE, 2016.

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