Introducing
∇*Sim*: Differentiable Simulation for Self-Supervised Parameter Estimation from Video

Anonymous

Introduction

What we see in a video is governed by the underlying 3D structure of the scene being captured, and the physical properties that determine its dynamics. Several of these properties are difficult to estimate from video without explicitly modeling the principles of motion and image formation. We tackle this ill-posed problem of reconstructing physical and geometric properties of objects from video, using *differentiable simulation.* To this end, we built ∇*Sim* ("gradSim"), a versatile end-to-end differentiable simulator, that enables backpropagation from video pixels to the physical and geometric parameters which generate them. We derive a general physical simulation framework that applies a discrete-time adjoint method to support a broad range of differentiable models, including 3D rigid bodies, solid and thin-shell deformable bodies, and incompressible fluids. All physical simulations can be differentiably rendered to generate videos, enabling the estimation of physical attributes and control parameters from video supervision alone. ∇*Sim* can solve a variety of inverse simulation and visual model-predictive control tasks, without relying on *state-based* supervision such as object velocities/positions.

∇*Sim* - Differentiable simulation

Typically, differentiable physics simulation [12, 10, 2, 3] and rendering [4, 7, 1, 6, 8] have been treated as mutually exclusive tasks. Here, we take on a unified view of simulation in general, to mean physics simulation *and* rendering. ∇*Sim* comprises two major components (*cf.* Fig. 1): a *differentiable physics engine* that computes the physical states of the world at each time instant, and a *differentiable renderer* that renders a 2D image.

Our **differentiable physics engine** computes the *state* of the world at each instant by integrating a series of time-stepping equations. For accurate physical simulations, a naive implementation using reverse mode automatic differentiation entails tracking the history of several thousand states for each second of simulation. To circumvent this, we instead consider our physics engine as a differentiable operation that provides an implicit relationship between a state vector at the start of a time step, and the updated state at the end of the time step. By application of the implicit function theorem, and using a source translation approach, we design an automatically differentiable physics engines that allows us to implement simulation *kernels* in a subset of python syntax, and computes the adjoints of each kernel at runtime, generating C++/CUDA [5] code. Kernels are wrapped as custom autograd operations on PyTorch tensors, which allows users to focus on the definition of physical models, and leverage the PyTorch tape-based autodiff to track the overall program flow.

Through this we are able to simulate: (a) Deformable Solids (a finite element model (FEM) with a stable Neo-Hookean constitutive model [9]), (b) Deformable Thin-Shells (a Neo-Hookean FEM with additional lift/drag forces), (c) Rigid Bodies (through Newton-Euler equations), and (d) contacts (relaxed friction [11] model).

Our **differentiable renderer** expects scene descriptions as input and generates color images as output, all according to a sequence of image formation stages defined by the forward graphics pipeline. The scene description includes a complete geometric descriptor of scene elements, their associated material properties, light sources, and camera parameters. The rendering process is not generally differentiable, as visibility and occlusion events introduce discontinuities. Our experiments employ two differentiable alternatives to traditional rasterization, SoftRas [7] and DIB-R [1], both of which rely on smoothing triangle edges by replacing their discontinuities with sigmoids. This has the effect of blurring triangle edges into semi-transparent boundaries, thereby removing the non-differentiable discontinuity of traditional rasterization.

∇*Sim* performs differentiable physics and rendering at independent and adjustable rates, allowing us to trade computation for accuracy by rendering fewer frames than physics simulation updates.

Experiments

In our **physical parameter estimation from video** experiments, we presented our algorithm with a video of an object with forces applied to it (e.g. objects falling, sliding, bouncing). When simulating rigid bodies, we estimated the mass, elasticity, or friction of the objects by comparing the video produced by our differentiable pipeline with the reference frames with an L2 loss. Compared to several baselines (e.g. a non-differentiable simulator approximating the gradient through REINFORCE, a CNN directly predicting the object properties, and an oracle with access to the dynamic states of the system), we found our method only beat by the oracle. During these experiments, we noticed that the loss landscape of our pipeline allowed for smooth and reliable optimization to the one local and global optimum (except for very small property values which lead to instabilities in any simulator), compared to the loss landscape obtained from the non-differentiable simulation, where the correct solution is in a local, not in the global optimum. We also ran experiments to estimate soft body properties (masses, elasticities, etc.), and we were able to achieve competitive results close to the oracle, despite only having video frames as input. We also ran several **visuomotor control** experiments, where the learning algorithm is presented with a final frame of a deformable object in a target position and orientation and we train a neural network controller to actuate the object and move it to the target. Figure 2 shows how our technique manages to reach the desired pose in most cases.

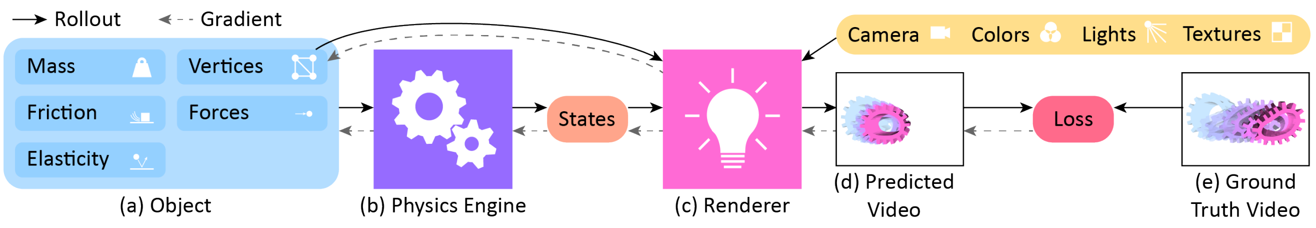

Figure 1: ∇***Sim*:** Given video observations of an evolving physical system (e), we randomly initialize scene object properties (a) and evolve them over time using a differentiable physics engine (b), which generates *states*. Our renderer (c) processes states, object vertices and global rendering parameters to produce image frames for computing our loss. We back-propagate to object shape and physical properties to refine our estimates.

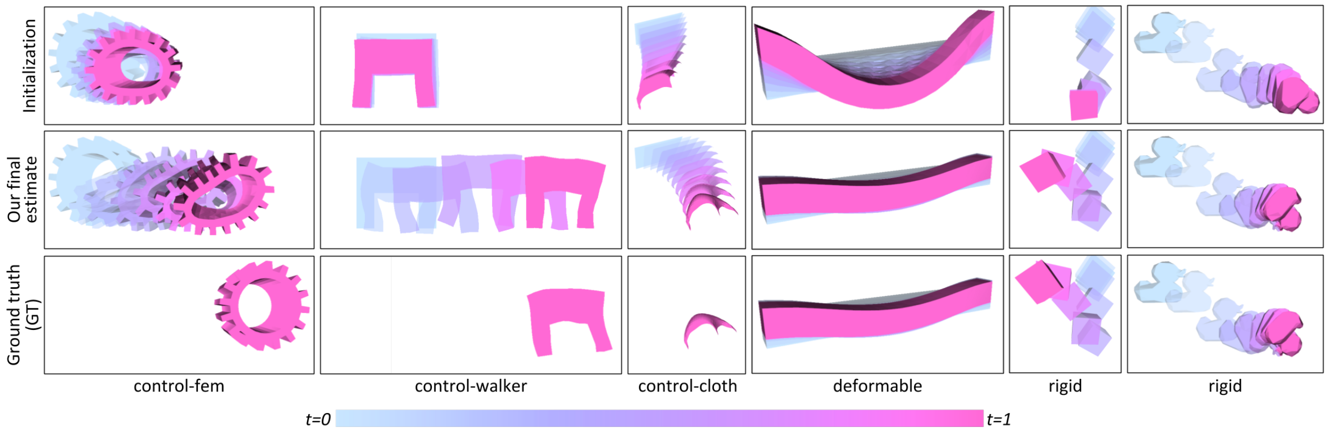

Figure 2: **Qualitative results**: ∇*Sim* accurately estimates physical parameters for diverse, complex environments. For `control-fem` and `control-walker` experiments, we train a neural network to actuate a soft body towards a target *image* (GT). For `control-cloth`, we optimize the cloth's initial velocity to hit a target pose (GT), all in under nonlinear lift/drag forces. For `deformable` experiments, we optimize the material properties of a beam to match a video. In the `rigid` experiments, we estimate contact parameters (elasticity/friction) and object density to match a *video* (GT). We visualize entire time sequences (t) with color-coded blends.

References

[1] Wenzheng Chen et al. "Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer". In: *NeurIPS* (2019).

[2] Jonas Degrave et al. "A Differentiable Physics Engine for Deep Learning in Robotics". In: *CoRR* abs/1611.01652 (2016). arXiv: `1611.01652`. URL: `http://arxiv.org/abs/1611.01652`.

[3] Yuanming Hu et al. "DiffTaichi: Differentiable Programming for Physical Simulation". In: *ICLR* (2020).

[4] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3D Mesh Renderer". In: *CVPR*. 2018.

[5] David Kirk et al. "NVIDIA CUDA software and GPU parallel computing architecture". In: *ISMM*. Vol. 7. 2007, pp. 103–104.

[6] Tzu-Mao Li et al. "Differentiable Monte Carlo Ray Tracing through Edge Sampling". In: *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* (2018).

[7] Shichen Liu et al. "Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning". In: *ICCV* (2019).

[8] Merlin Nimier-David et al. "Mitsuba 2: A Retargetable Forward and Inverse Renderer". In: *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* (2019).

[9] Breannan Smith, Fernando De Goes, and Theodore Kim. "Stable neo-hookean flesh simulation". In: *ACM Transactions on Graphics (TOG)* 37.2 (2018), pp. 1–15.

[10] Changkyu Song and Abdeslam Boularias. *Learning to Slide Unknown Objects with Differentiable Physics Simulations*. 2020. arXiv: `2005.05456 [cs.RO]`.

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

[12] Marc Toussaint et al. "Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning". In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, June 2018. DOI: `10.15607/RSS.2018.XIV.044`.