Learning to Control PDEs with Differentiable Physics

Seminar – Deep Learning in Physics

TUM, Summer Semester 2022

Barnabás Börcsök

Advisor: Lukas Prantl

Overview

- Differentiable Physics
  - Motivation
  - Forward Physics Simulation
  - Differentiable *(“backwards”)* Physics
  - Examples
- Partial Differential Equations (PDEs)
- Results of the paper
- Other examples and outlook
  - Enhancing simulations with Neural Networks (NNs)
  - Inductive Physical Bias

Overview

- Differentiable Physics
  - Motivation
  - Forward Physics Simulation
  - Differentiable *(“backwards”)* Physics
  - Examples
- Partial Differential Equations (PDEs)
- Results of the paper
- Other examples and outlook
  - Enhancing simulations with Neural Networks (NNs)
  - Inductive Physical Bias

The big picture – Computer Graphics

Virtual world

Real world

The big picture – Computer Graphics

- Modelling
- Simulating
- Rendering

… the real world

Virtual world

Real world

The big picture – Computer Graphics

- Modelling
- Simulating
- Rendering

Virtual world

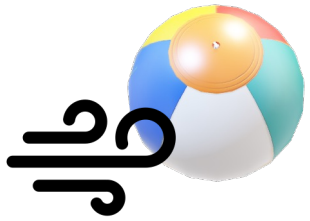
Real world

The big picture – Computer Graphics

- Modelling
- Simulating
- Rendering

Virtual world

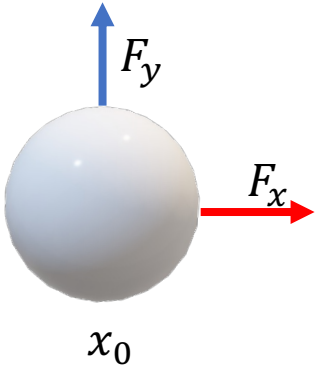
Real world



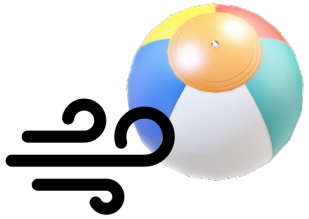
The big picture – Computer Graphics

- Modelling
- Simulating
- Rendering

Virtual world



Real world

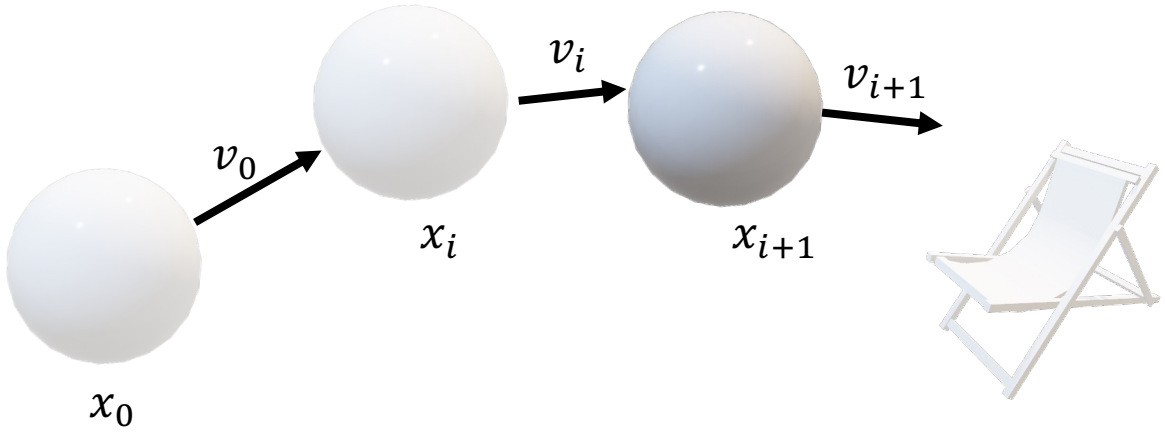


The big picture – Computer Graphics

- Modelling
- Simulating
- Rendering

$$(x_{i+1}, v_{i+1}) = f(x_i, v_i)$$

Virtual world



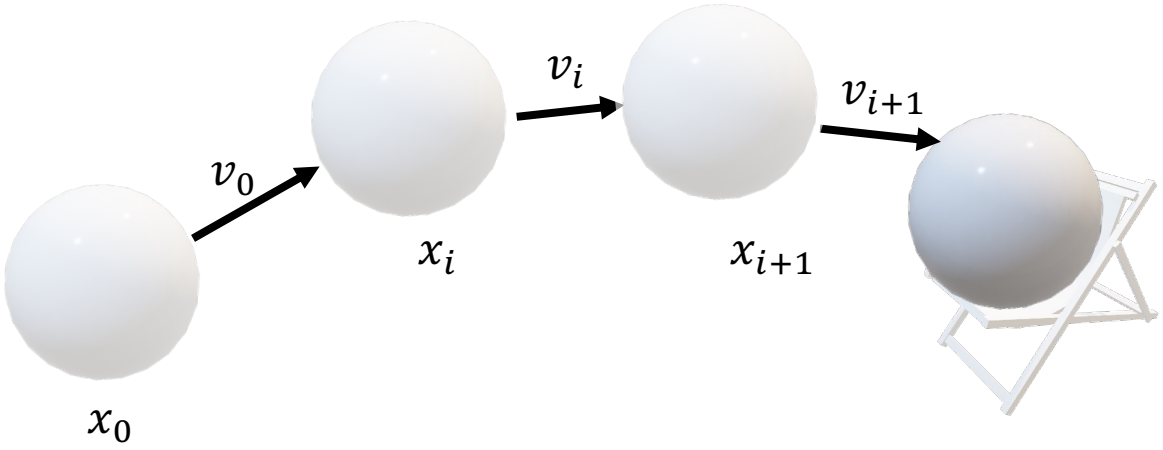
Real world

The big picture – Computer Graphics

- Modelling
- Simulating
- Rendering

$$(x_{i+1}, v_{i+1}) = f(x_i, v_i)$$

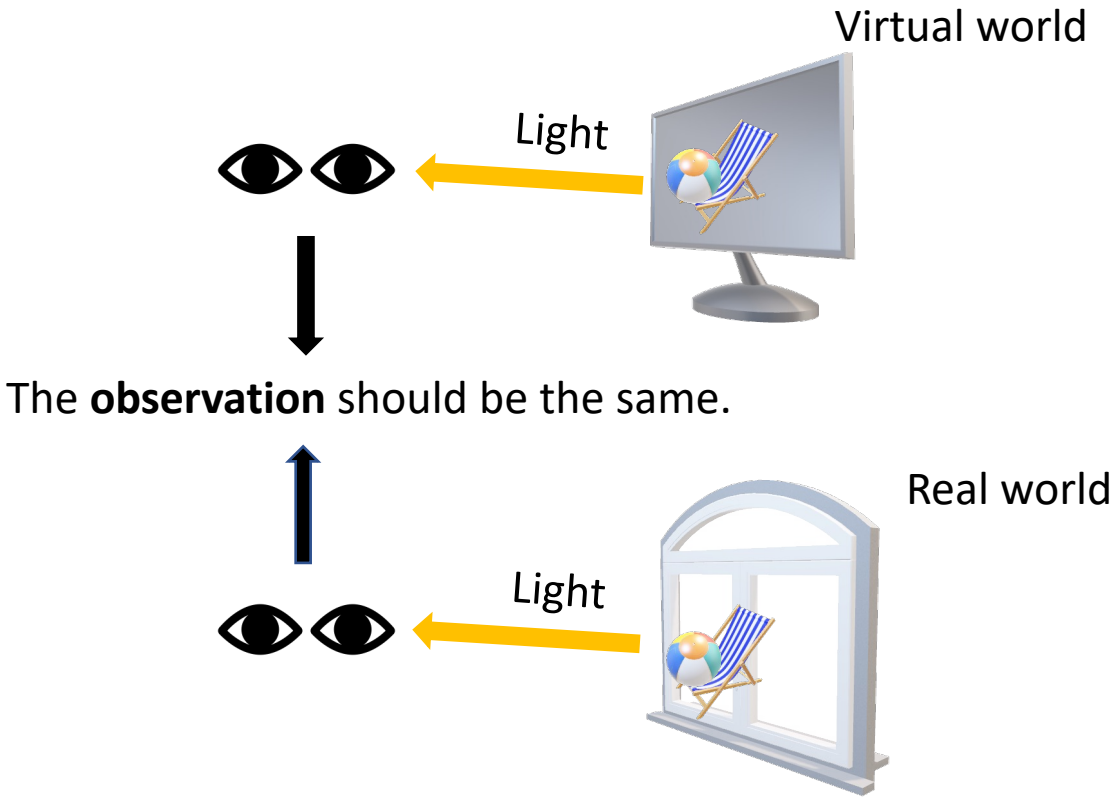
Virtual world



Real world

The big picture – Computer Graphics

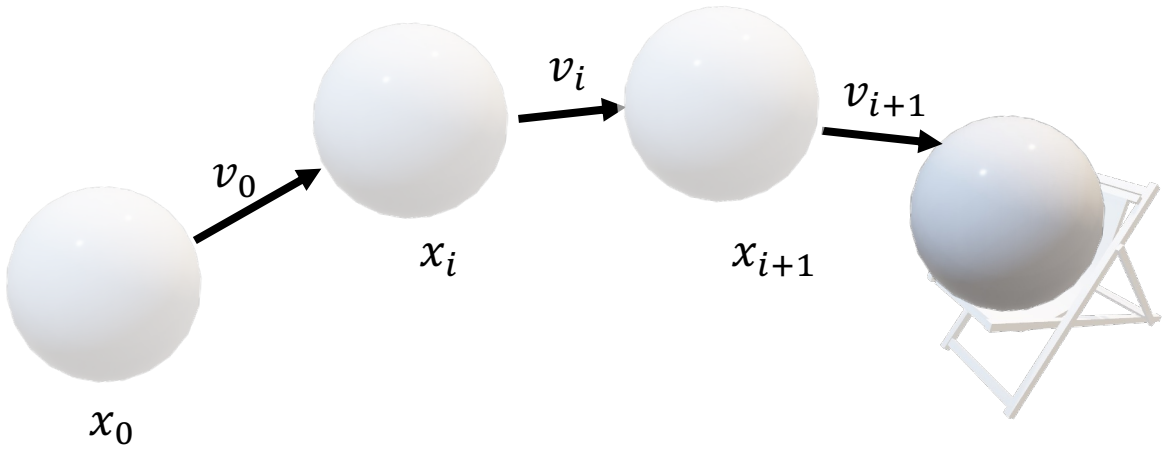
- Modelling
- Simulating
- Rendering



(Forward) Physics Simulation

Physical simulations are much more complex …

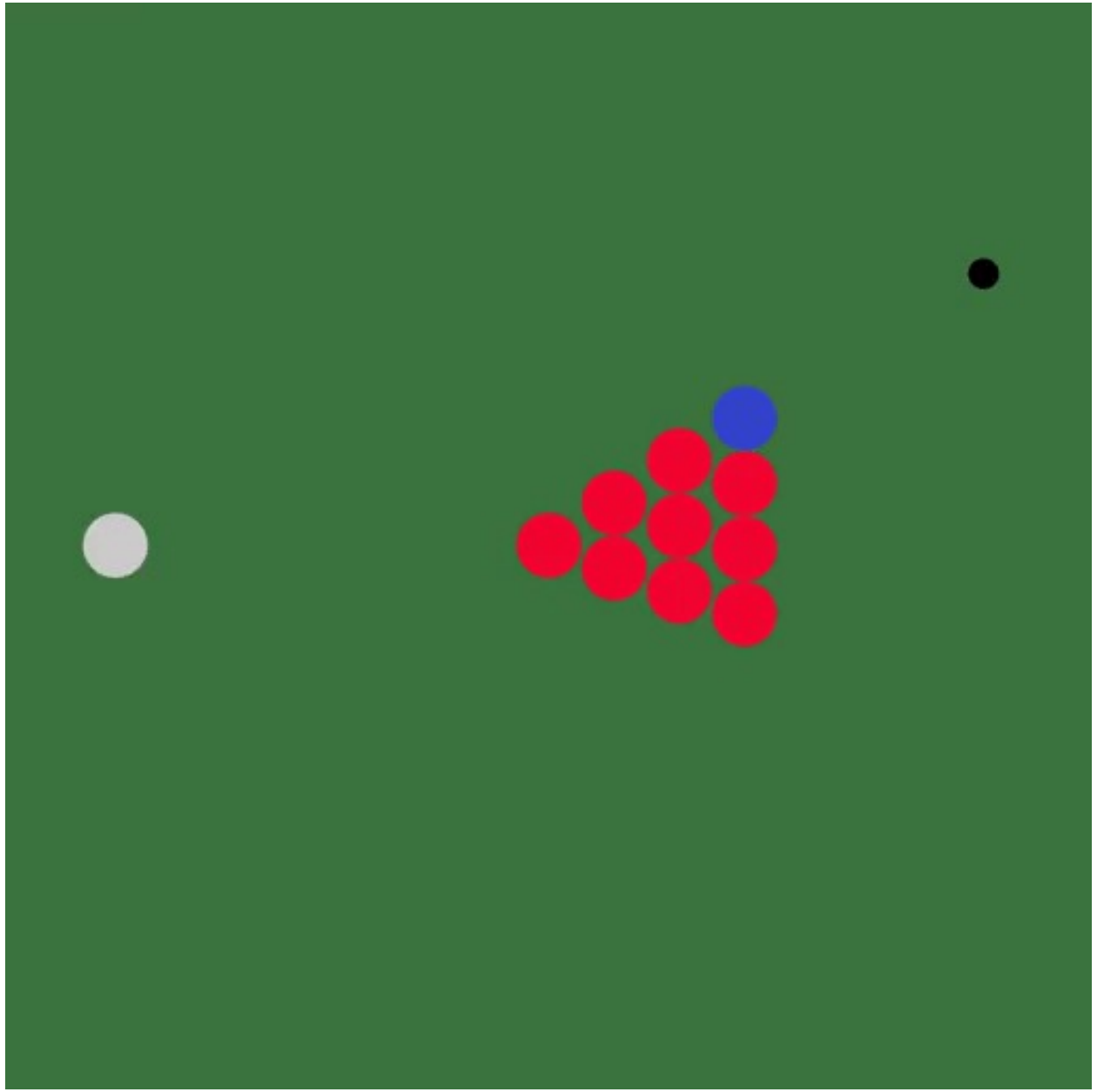
$$(x_{i+1}, v_{i+1}) = f(x_i, v_i)$$



Our Model

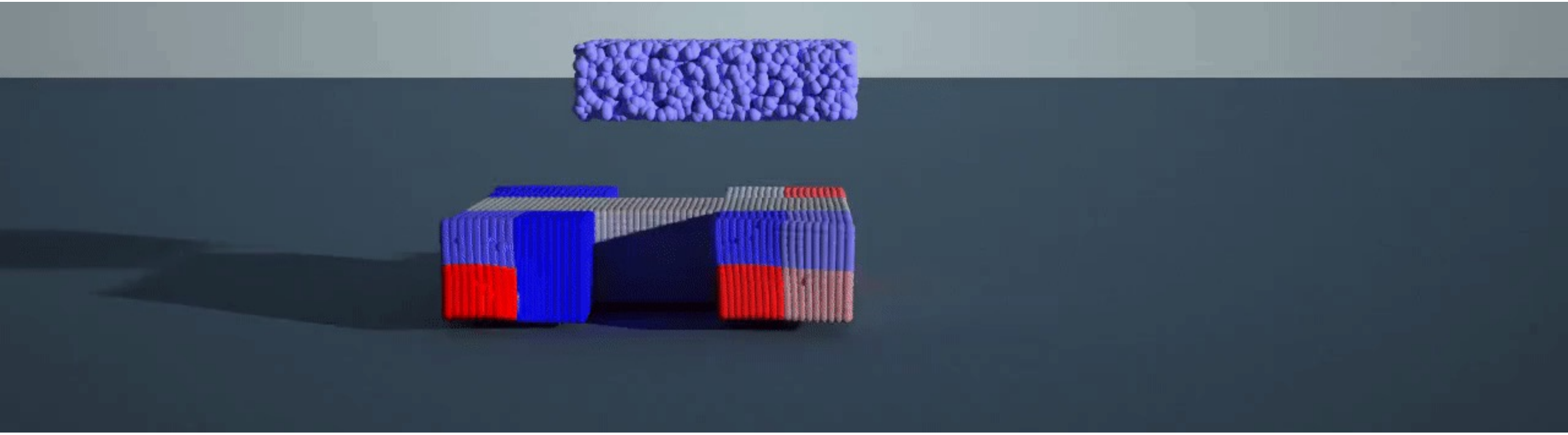
Artistic Simulation of Curly Hair, Hayley et. al

Each strand is made up of springs (Hooke’s law)
Simulating 579 hair strands (44,552 points)

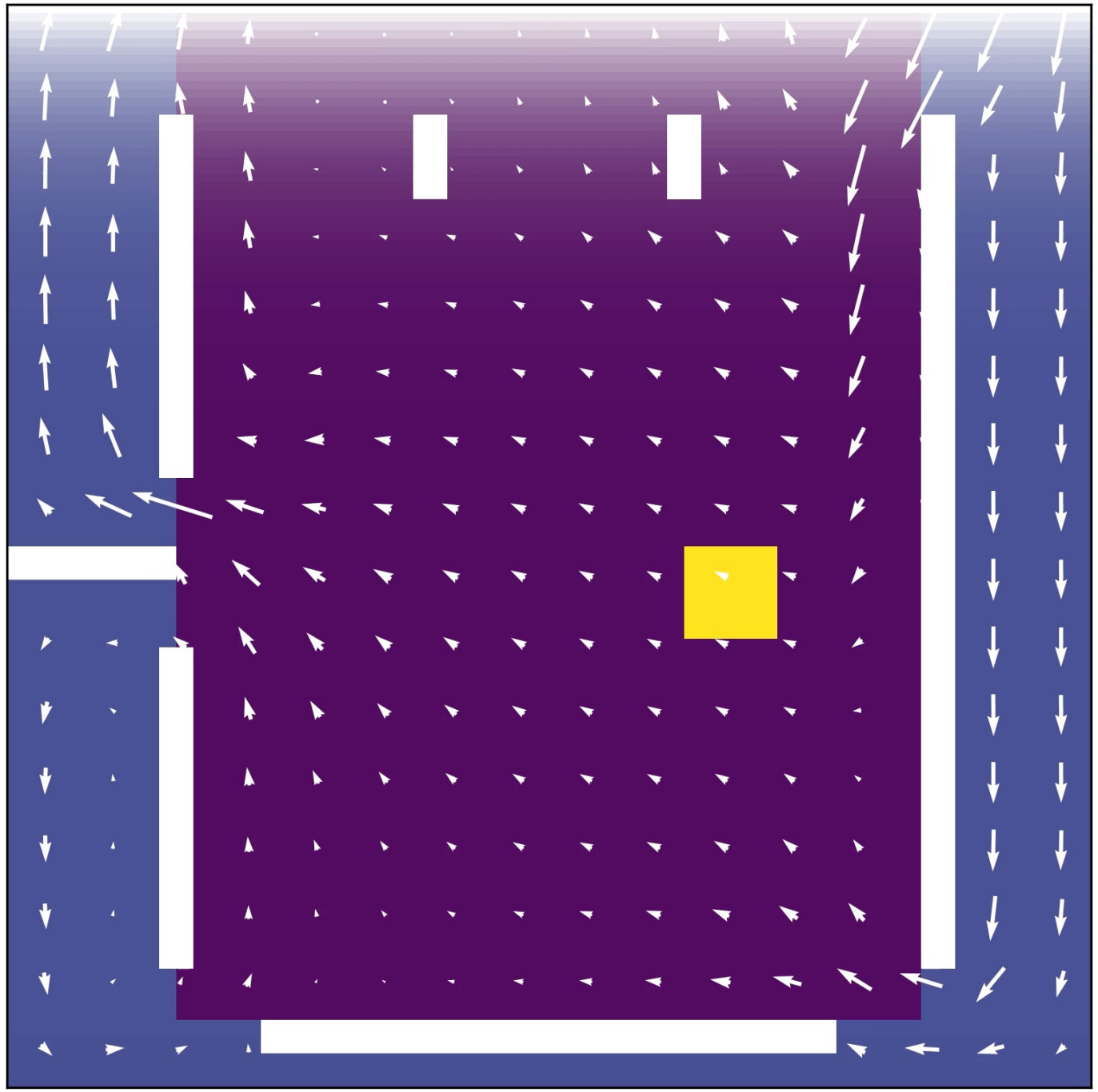


Source: https://github.com/taichi-dev/difftaichi

Elastic Object + Fluid



Source: https://github.com/taichi-dev/difftaichi

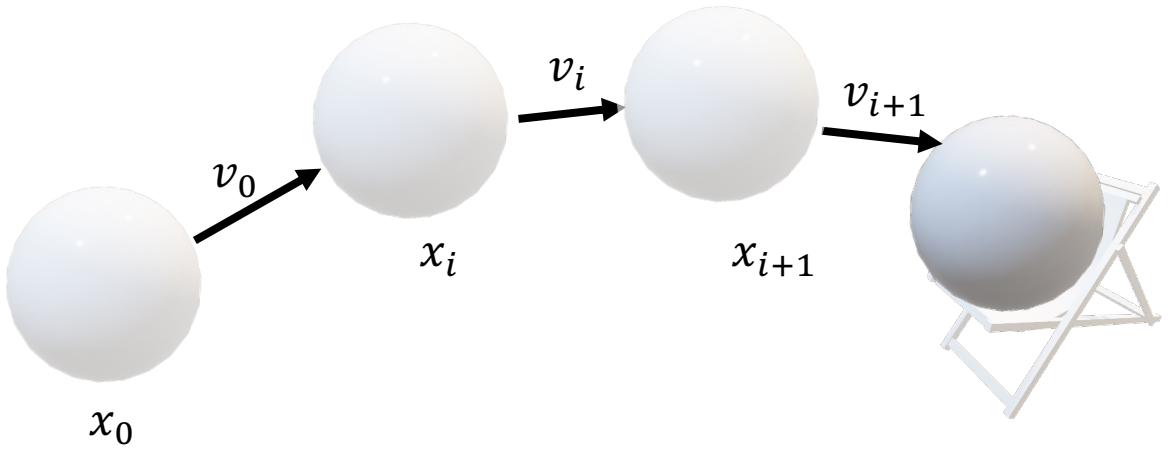


2D Indirect Fluid Control

Source:
Learning to Control PDEs with Differentiable Physics,
Holl et. al

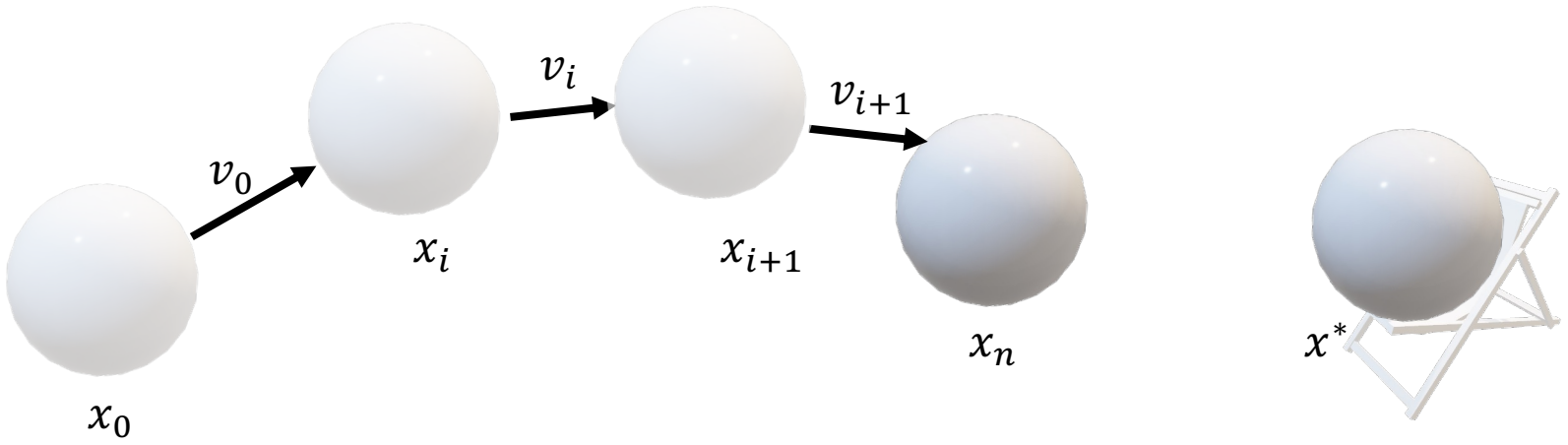
Differentiable Physics Simulation

$$(x_{i+1}, v_{i+1}) = f(x_i, v_i)$$



Differentiable Physics Simulation

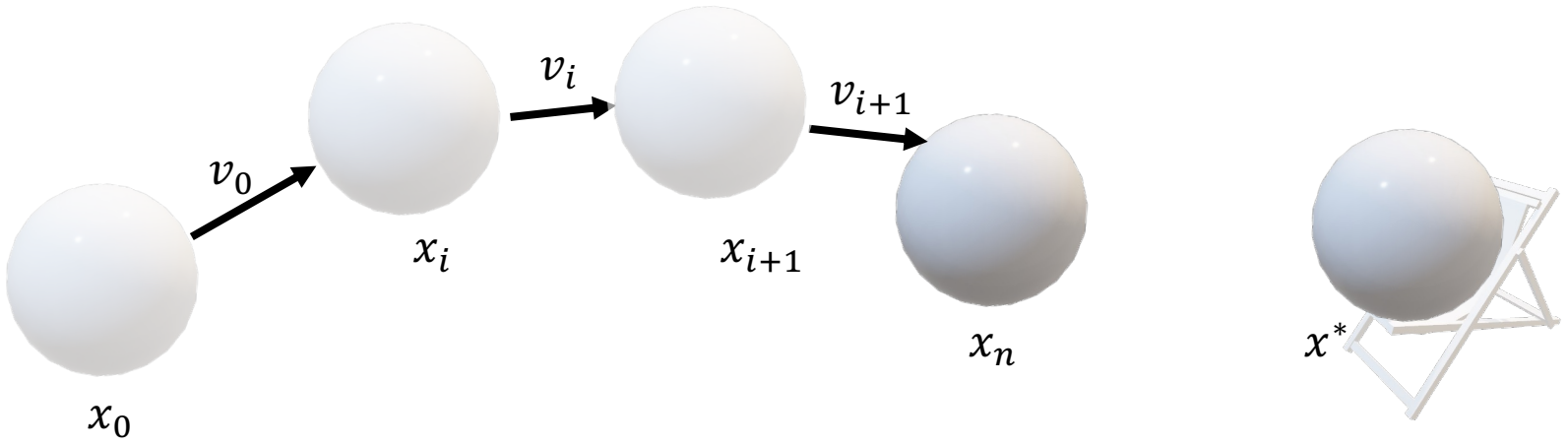
$L(x_n) = |x^* - x_n|_2^2$ "Loss function"



Differentiable Physics Simulation

$L(x_n) = |x^* - x_n|_2^2$ "Loss function"

Let's compute $\frac{\partial L}{\partial x_0}$ and $\frac{\partial L}{\partial v_0}$.

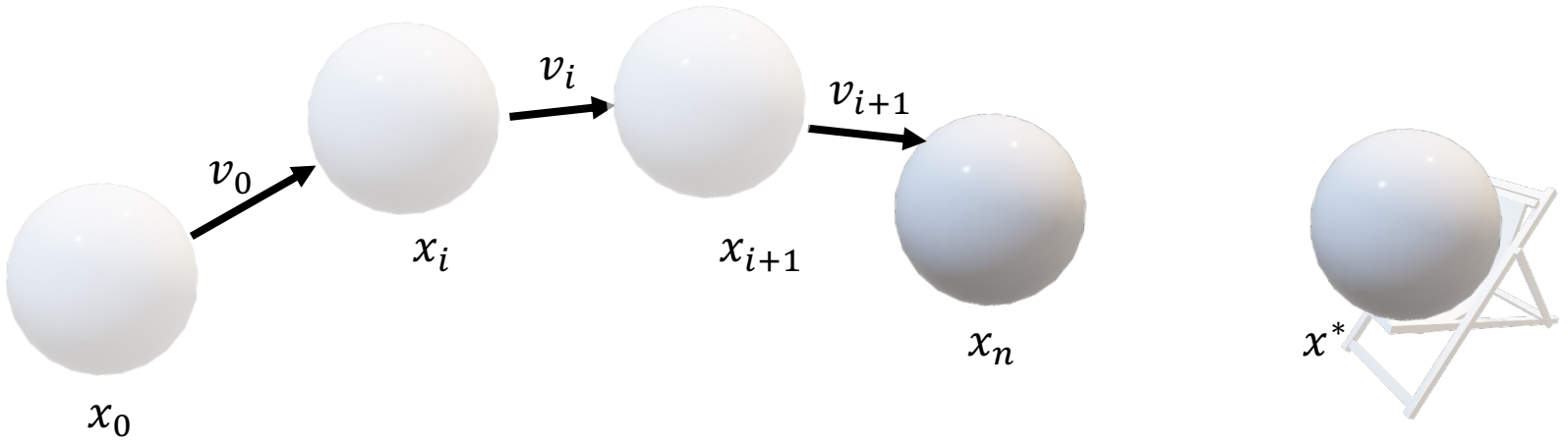


Differentiable Physics Simulation

$L(x_n) = |x^* - x_n|_2^2$ "Loss function"

Let's compute $\frac{\partial L}{\partial x_0}$ and $\frac{\partial L}{\partial v_0}$.

Using the chain rule, a.k.a. "backpropagation"

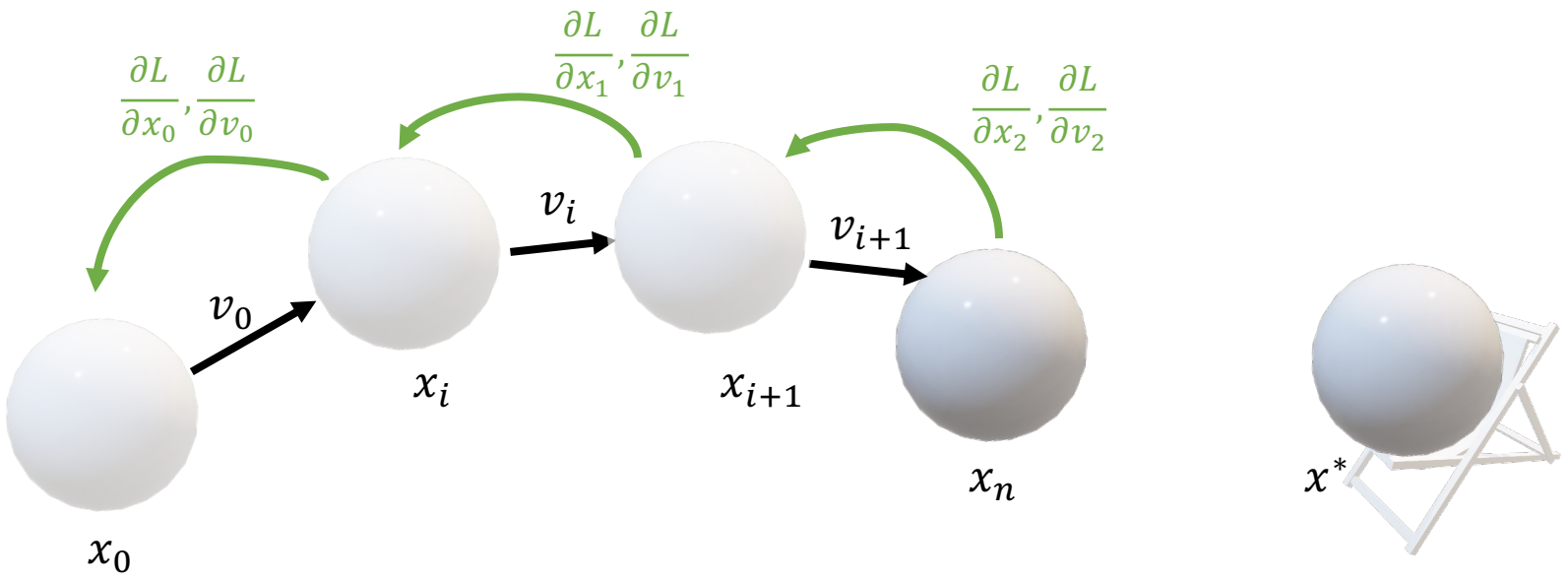


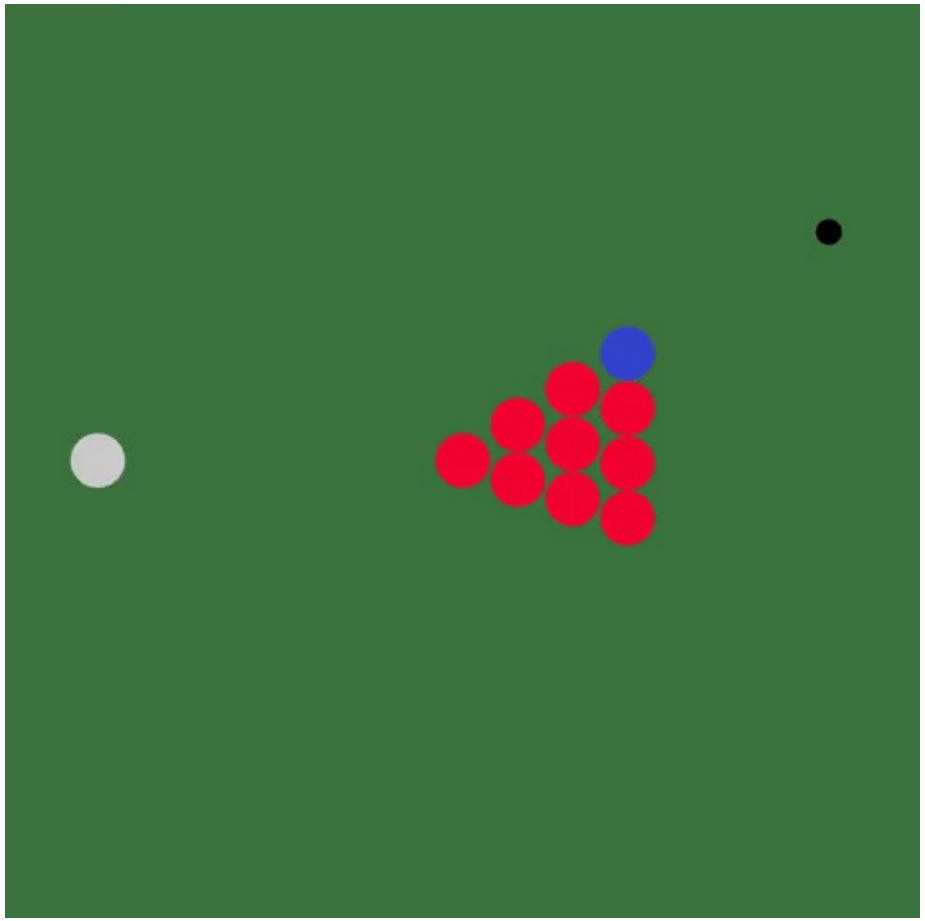
Differentiable Physics Simulation

$L(x_n) = |x^* - x_n|_2^2$ "Loss function"

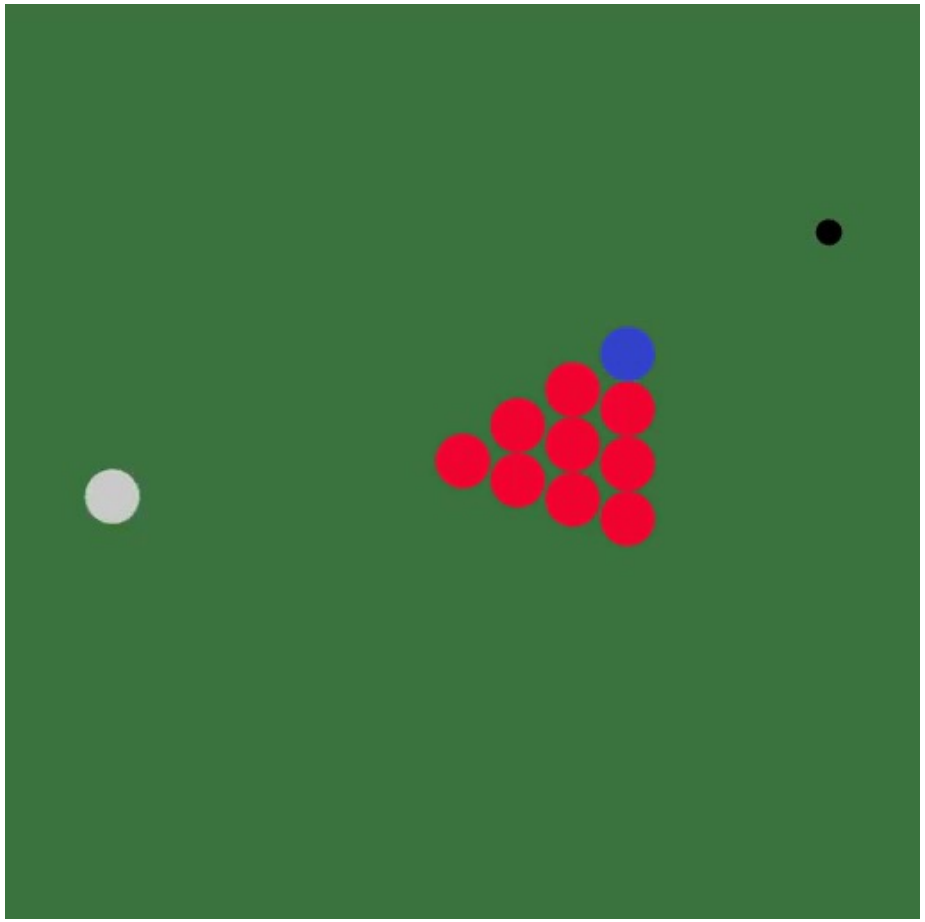
Let's compute $\frac{\partial L}{\partial x_0}$ and $\frac{\partial L}{\partial v_0}$.

Using the chain rule, a.k.a. "backpropagation"





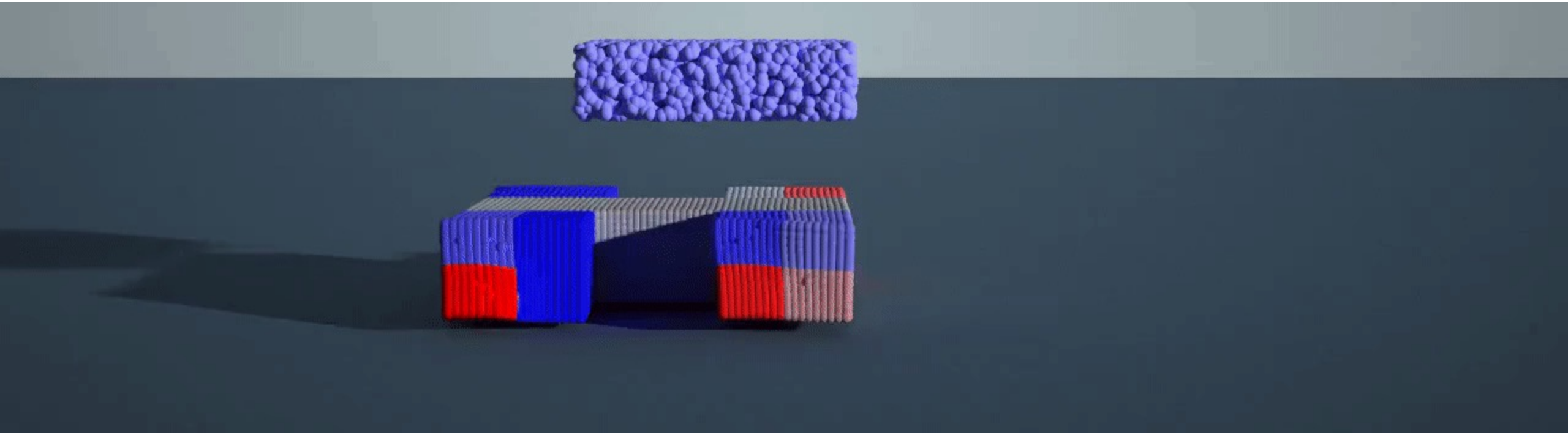
Gradient descent iteration 0 …



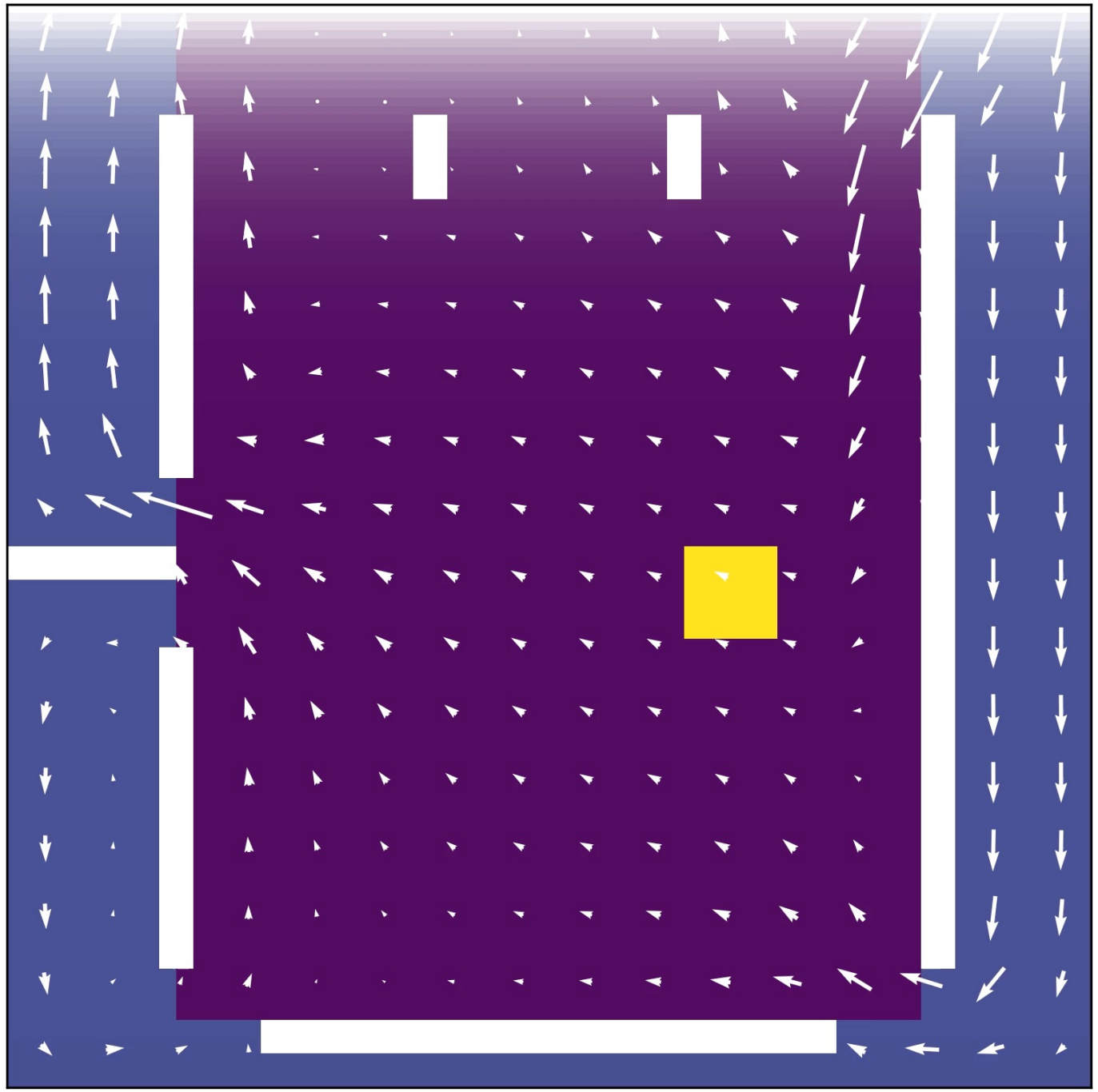
… and iteration 100

Source: https://github.com/taichi-dev/difftaichi#differentiable-billiard-simulator-python3-billiardspy

Elastic Object + Fluid
450 gradient descent iteration



Source: https://github.com/taichi-dev/difftaichi



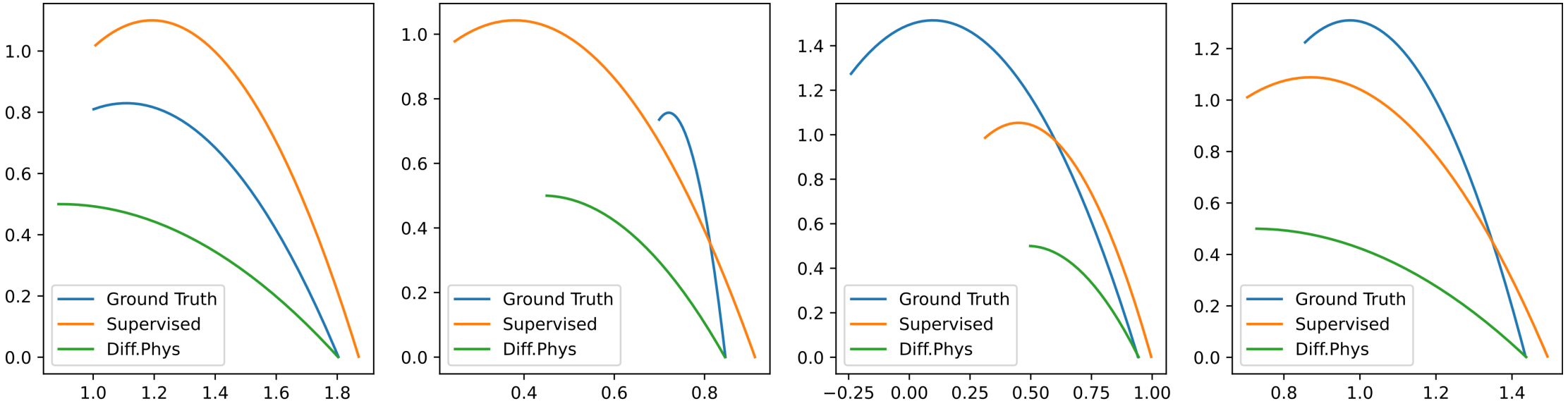
2D Indirect Fluid Control

100 training samples, each with 16 time steps

Learns to move >99% of the volume into the bucket

Source:
Learning to Control PDEs with Differentiable Physics,
Holl et. al

Supervised vs. Diff. Physics loss



Learning to throw, first 4 test cases

Overview

- Differentiable Physics
  - Motivation
  - Forward Physics Simulation
  - Differentiable *(“backwards”)* Physics
  - Examples
- Partial Differential Equations (PDEs)
- Results of the paper
- Other examples and outlook
  - Enhancing simulations with Neural Networks (NNs)
  - Inductive Physical Bias

Partial Differential Equations (PDEs)

- The most fundamental description of evolving systems
- From quantum mechanics to turbulent flows
- For a physical system $\boldsymbol{u}(\boldsymbol{x}, t)$

$$\frac{\partial \boldsymbol{u}}{\partial t} = \mathcal{P}\left(\boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}, \frac{\partial^2 \boldsymbol{u}}{\partial \boldsymbol{x}^2}, ..., \boldsymbol{y}(t)\right)$$

$$\frac{\partial \boldsymbol{u}}{\partial t} = \mathcal{P}\left(\boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}, \frac{\partial^2 \boldsymbol{u}}{\partial \boldsymbol{x}^2}, ...\right) + \boldsymbol{F}(t)$$

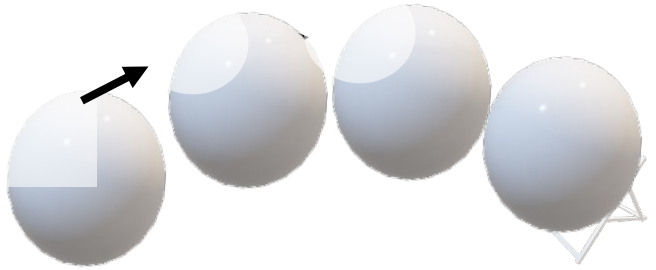
- Agent only has access to $\boldsymbol{o}(\boldsymbol{u})$ (e.g. density, but not velocity)

Partial Differential Equations (PDEs)

$$\frac{\partial \boldsymbol{u}}{\partial t} = \mathcal{P}\left(\boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}, \frac{\partial^2 \boldsymbol{u}}{\partial \boldsymbol{x}^2}, \ldots\right) + \boldsymbol{F}(t)$$

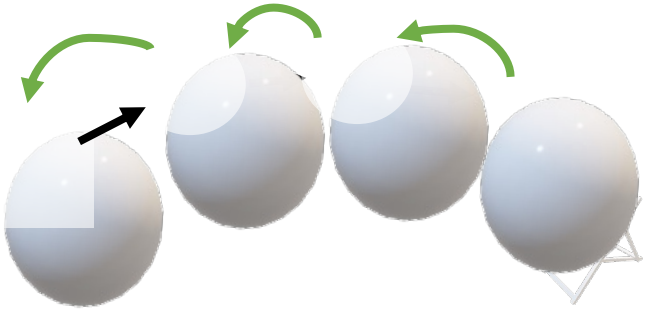
- A Solver moves the system forward by a time increment Δt

$$\boldsymbol{u}(t_{i+1}) = \text{Solver}[\boldsymbol{u}(t_i), \boldsymbol{y}(t_i)] = \boldsymbol{u}(t_i) + \Delta t \cdot \mathcal{P}\left(\boldsymbol{u}(t_i), \ldots, \boldsymbol{y}(t_i)\right)$$



- Differentiable Solvers can efficiently compute the derivatives with respect to any of the inputs:

$$\partial \boldsymbol{u}(t_{i+1}) / \partial \boldsymbol{u}(t_i) \text{ and } \partial \boldsymbol{u}(t_{i+1}) / \partial \boldsymbol{y}(t_i)$$



Goal

i. Find optimal trajectory

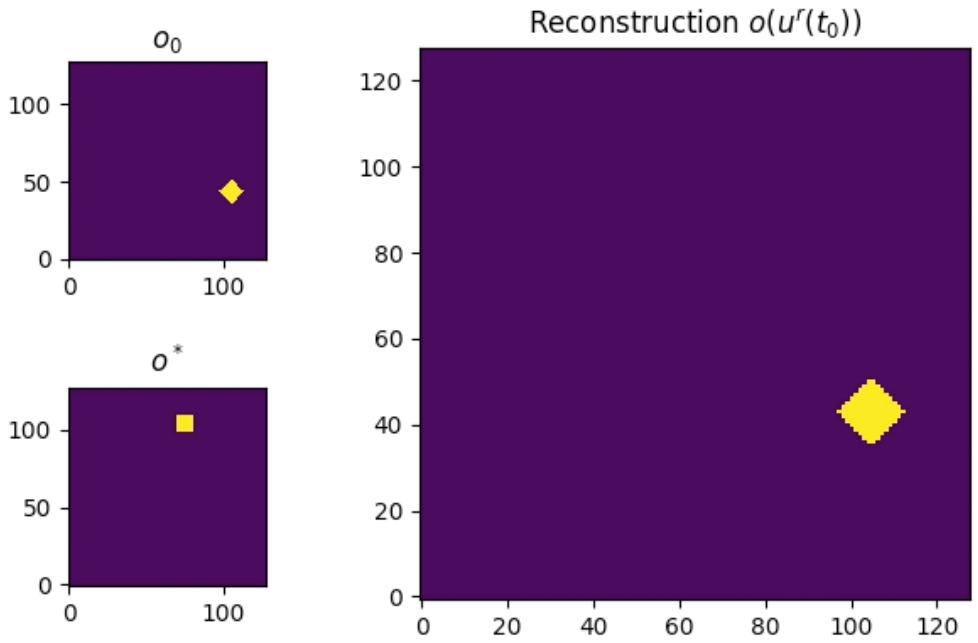
   Observation Predictor (OP) agent

ii. Move the system along it

   Corrector Network agent:

   Control Force Estimator (CFE)

(+ minimize the force applied)

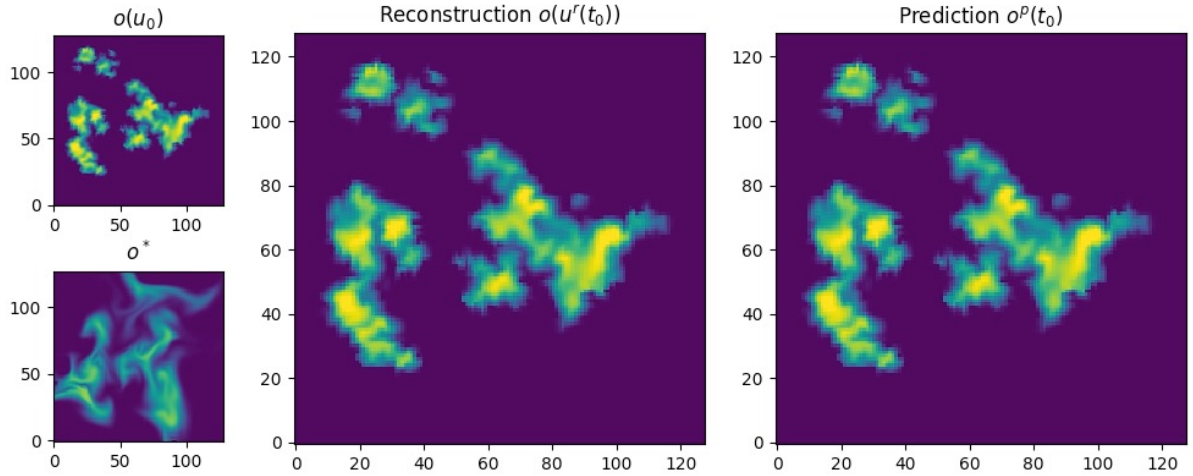


Results

1. Burger’s Equation
2. Incompressible Fluid Flow
    1. Natural fluid flow reconstruction
    2. Shape transition
3. Incompressible Fluid with Indirect Control

Results

1. Burger's Equation
2. Incompressible Fluid Flow
    1. Natural fluid flow reconstruction
    2. Shape transition
3. Incompressible Fluid with Indirect Control



Overview

- Differentiable Physics
  - Motivation
  - Forward Physics Simulation
  - Differentiable *(“backwards”)* Physics
  - Examples
- Partial Differential Equations (PDEs)
- Results of the paper
- Other examples and outlook
  - Enhancing simulations with Neural Networks (NNs)
  - Inductive Physical Bias

Neural Networks are Universal Function Approximators

- Numbers in, Numbers out

"AI" → "It's a squirrel"

$$VGG-16\left(\begin{bmatrix}(0.8,0.6,0.15) & \cdots & (0.87\ 0.4,0.2)\\ \vdots & \ddots & \vdots \\ (0.3,0.5,0.4) & \cdots & (0.3,0.7,0.2)\end{bmatrix}\right) = [.02,\ldots,.001,0.4221,\ldots]$$

Neural Networks are Universal Function Approximators

- Pros:
  - Fast
- Con
  - “black box”, hard to explain
  - Unreliable outside known domain
  - Training

Neural Networks are Universal Function Approximators

- Pros:
  - Fast
- Con
  - “black box”, hard to explain
  - Unreliable outside known domain
  - Training

Idea: Let’s combine the “fast” parts and the “reliable” parts

Neural Networks are Universal Function Approximators

- Pros:
  - Fast
- Con
  - “black box”, hard to explain
  - Unreliable outside known domain
  - Training

Idea: Let’s combine the “fast” parts and the “reliable” parts

Inductive Physical Bias

Inductive Physical Bias

- Assumptions used while learning
- Underlying physical system

Inductive Physical Bias

- Assumptions used while learning
- Underlying physical system

$$\frac{\partial \boldsymbol{u}}{\partial t} = \mathcal{P}\left(\boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}, \frac{\partial^2 \boldsymbol{u}}{\partial \boldsymbol{x}^2}, \ldots\right) + \boldsymbol{F}(t)$$

Deep Learning for Cloud Rendering

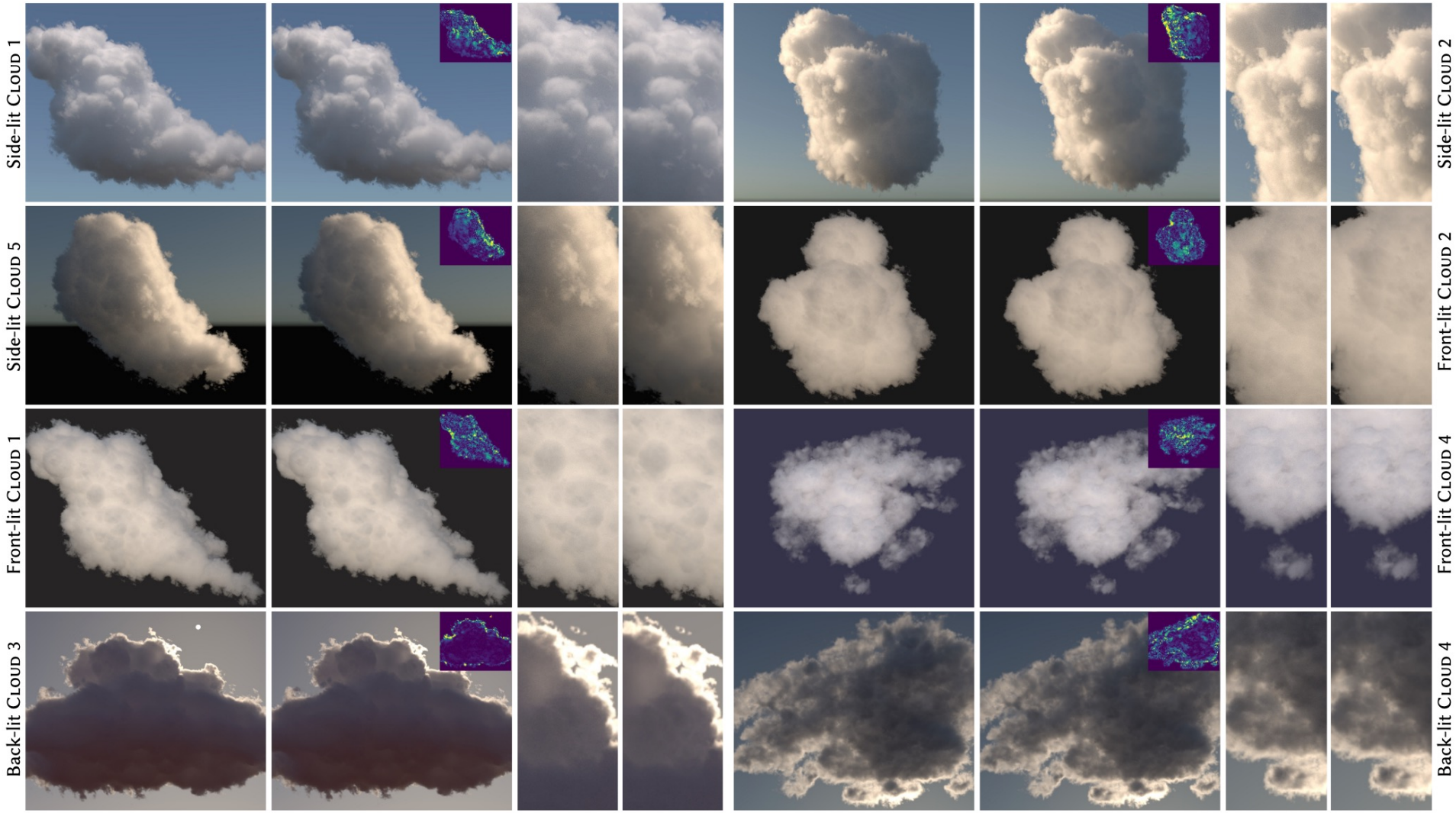
- Approximating the Radiative Transfer Function

$$L(\mathbf{x}, \omega) = \int_0^b T(\mathbf{x}, \mathbf{x}_u)\mu_s(\mathbf{x}_u) \int_{S^2} p(\omega \cdot \widehat{\omega}) L(\mathbf{x}_u, \widehat{\omega})\, \mathrm{d}\widehat{\omega}\, \mathrm{d}u + T(\mathbf{x}, \mathbf{x}_b) L(\mathbf{x}_b, \omega),$$

- By learning the indirect incoming radiance function

$$L_i(\mathbf{x}, \omega) = \int_{S^2} p(\omega \cdot \widehat{\omega})(L(\mathbf{x}, \widehat{\omega}) - L_d(\mathbf{x}, \widehat{\omega}))\, \mathrm{d}\widehat{\omega}.$$

Deep Learning for Cloud Rendering – Results



24x faster convergence than path tracing (PT) on average!

Main takeaway

- Don’t throw away existing knowledge!
- Leverage already existing methods
- Deep Learning **is not magic**, and shouldn’t be treated as such

Overview

- Differentiable Physics
  - Motivation
  - Forward Physics Simulation
  - Differentiable *(“backwards”)* Physics
  - Examples
- Partial Differential Equations (PDEs)
- Results of the paper
- Other examples and outlook
  - Enhancing simulations with Neural Networks (NNs)
  - Inductive Physical Bias

Sources

Papers:

- Learning to Control PDEs with Differentiable Physics [Holl et al. 2020] (+ the supplemental materials) https://ge.in.tum.de/publications/2020-iclr-holl/
- DiffTaichi: Differentiable Programming for Physical Simulation https://arxiv.org/abs/1910.00935
- Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks, by Kallweit et al. [2017] *https://la.disneyresearch.com/publication/deep-scattering/*

Talks:

- Differentiable Physics (for Deep Learning), Overview Talk by Nils Thuerey *[https://youtu.be/BwuRTpTR2Rg, 40m48s]*
- Differentiable Physical Simulation and AI @ NeurIPS 2020 DiffCVGP workshop *[https://youtu.be/i2O72iMe9ug, 29m59s]*
  - *Differentiable physics and the Taichi programming language*
  - *Code repo with examples: https://github.com/taichi-dev/difftaichi*