Differentiable Cloth Simulation for Inverse Problems

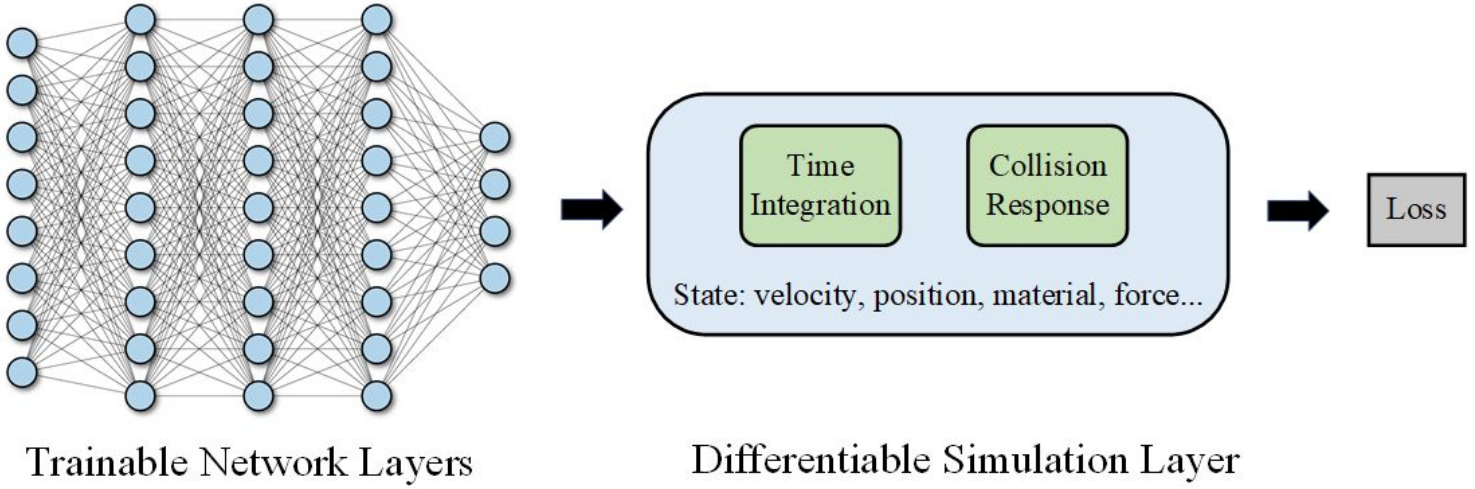
Junbang Liang

Content

- Motivation
- Related Work
- Our Method
  - Simulation pipeline
  - Gradient Computation
- Results

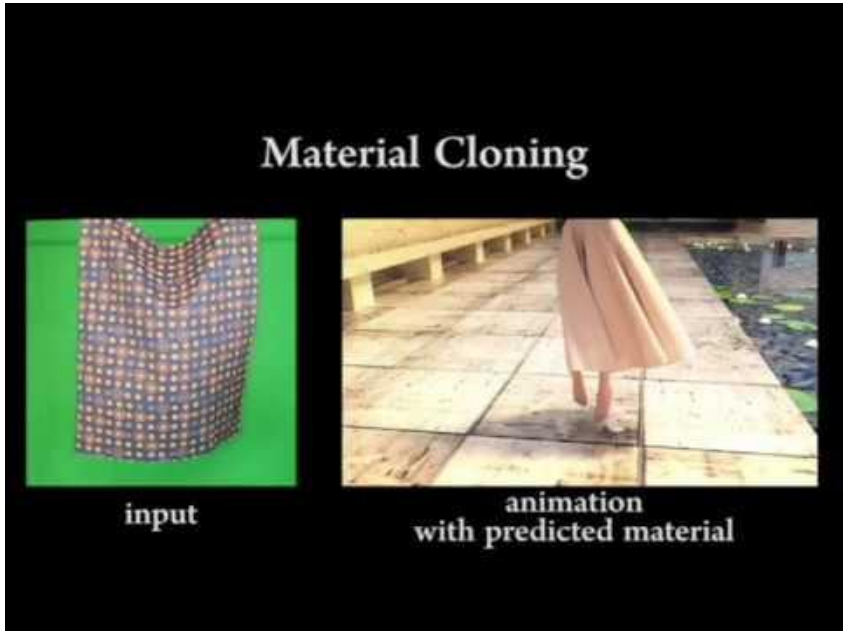
Motivation

- Differentiable Physics Simulation as a Network Layer
  - Physical property estimation
  - Control of physical systems

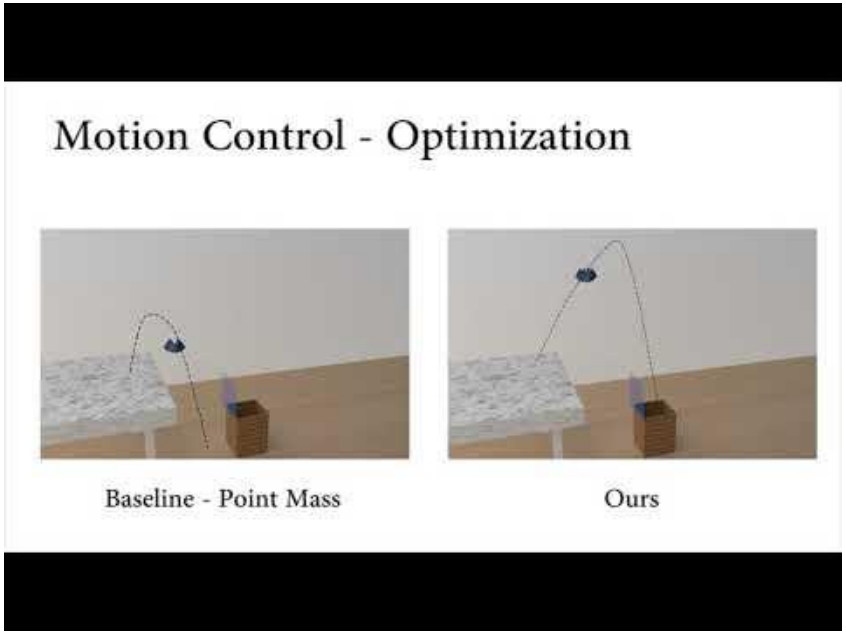


Motivation

- Differentiable Physics Simulation as a Network Layer
  - Physical property estimation
  - Control of physical systems



Yang et al. (2017)



Demo of our differentiable simulation

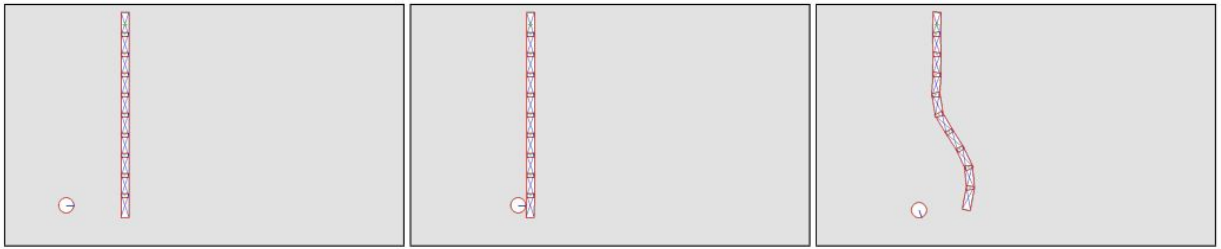
Content

- Motivation
- Related Work
- Our Method
  - Simulation pipeline
  - Gradient Computation
- Results

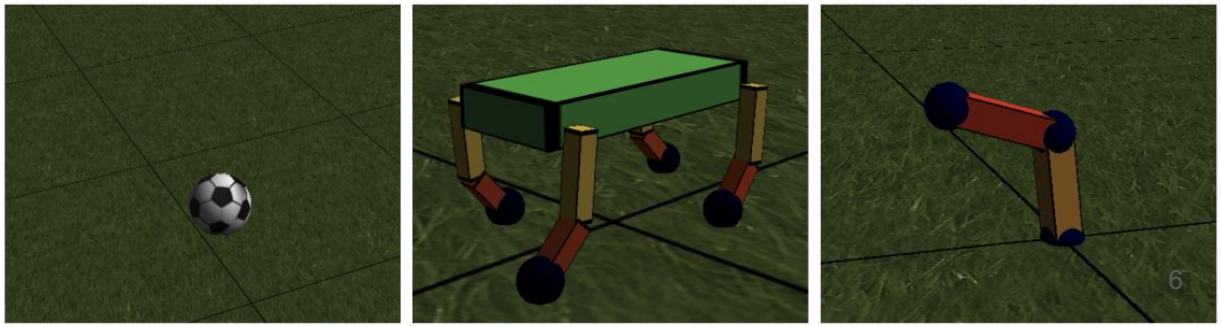
Related Work

- Differentiable rigid body simulation
  - Formulation not scalable to high dimensionality

Belbute-Peres et al. 2019

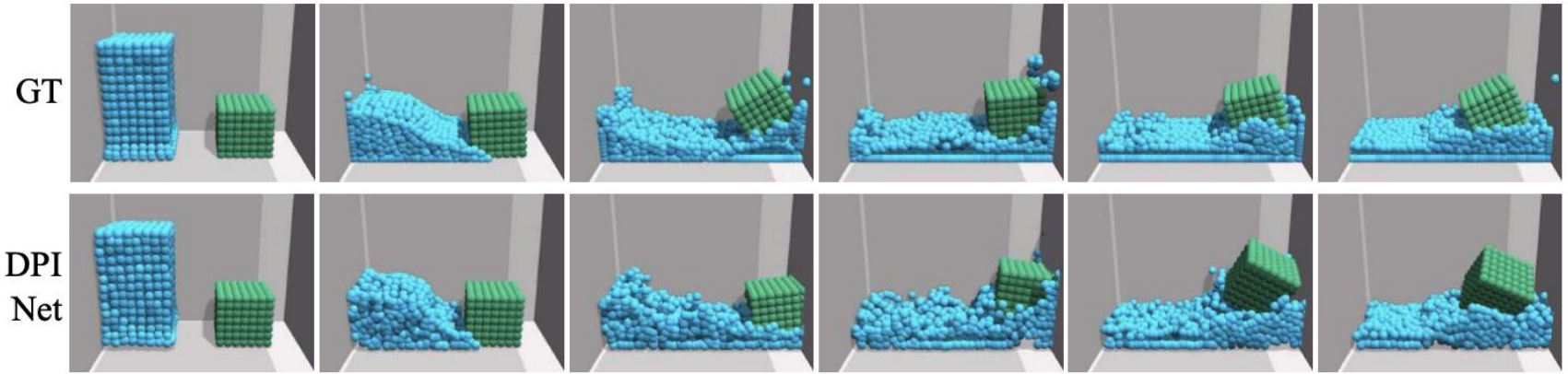


Degrave et al. 2019



Related Work

- Learning-based physics [Li et al. 2018]
  - Unable to guarantee physical correctness



Our Contributions

- Dynamic collision handling to reduce dimensionality
- Gradient computation of collision response using implicit differentiation
- Optimized backpropagation using QR decomposition

Content

- Motivation
- Related Work
- Our Method
  - Simulation pipeline
  - Gradient Computation
- Results

Introduction to Simulation

- Partial differential equation (PDE) of Newton's law:
- Solve $y(x,t)$ satisfying $\frac{\partial^2 y}{\partial t^2} = \frac{1}{\rho} f(y, \frac{\partial y}{\partial x})$, where $y(x,0) = y_0(x)$

- Discretization to ordinary differential equations (ODE):
- Solve $\mathbf{y}(t) = [y(x_i,t)]_i$ satisfying $\frac{d^2 y}{dt^2} = \frac{1}{m}\mathbf{f}(\mathbf{y}, \frac{\Delta \mathbf{y}}{\Delta \mathbf{x}})$, where $\mathbf{y}(0) = [y_0(x_i)]_i$

Introduction to Simulation

- Partial differential equation (PDE) of Newton's law:
- Solve $y(x,t)$ satisfying $\frac{\partial^2 y}{\partial t^2} = \frac{1}{\rho} f(y, \frac{\partial y}{\partial x})$, where $y(x,0) = y_0(x)$

- Discretization to ordinary differential equations (ODE):
- Solve $\mathbf{y}(t) = [y(x_i,t)]_i$ satisfying $\frac{d^2 y}{dt^2} = \frac{1}{m}\mathbf{f}(\mathbf{y}, \frac{\Delta \mathbf{y}}{\Delta \mathbf{x}})$, where $\mathbf{y}(0) = [y_0(x_i)]_i$

Point Cloud Simulation Flow

1. Init $\mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$
2. Compute $\Delta \mathbf{v}$ from $\mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$
   - Newton's method
3. $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1} * \Delta t, \mathbf{v}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$
4. $t = t + 1,$ goto 2

Cloth Simulation Flow

1. Init $\mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$
2. Compute $\Delta \mathbf{v}$ from $\mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$
   - Newton's method
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$
5. $t = t + 1,$ goto 2

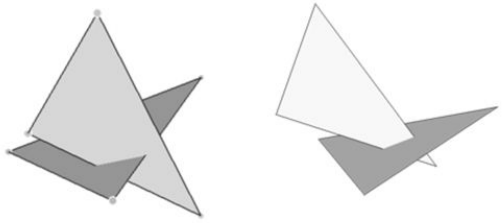
Collision Response

- Collision Detection: $dist(\text{node}_i, \text{face}_j, t) < \delta$, where δ is the cloth thickness, and t is some time between two steps.
- Objective: introduce minimum energy to avoid collision:

$$dist(\text{node}_i, \text{face}_j, t) - \delta \geq 0$$

$$\downarrow$$

$$C_{ee} = \boldsymbol{n} \cdot [(\alpha_3 \boldsymbol{x}_3 + \alpha_4 \boldsymbol{x}_4) - (\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2)]$$
$$C_{vf} = \boldsymbol{n} \cdot [\boldsymbol{x}_4 - (\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2 + \alpha_3 \boldsymbol{x}_3)]$$



EE Impact

VF Impact

Collision Response

- Collision Detection: $dist(\text{node}_i, \text{face}_j, t) < \delta$, where δ is the cloth thickness, and t is some time between two steps.
- Objective: introduce minimum energy to avoid collision:

$$dist(\text{node}_i, \text{face}_j, t) - \delta \geq 0$$

- Constraint formulation: $\mathbf{Gx} + \mathbf{h} \leq 0$
- Objective formulation: Quadratic Programming:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{z} - \mathbf{x})^\top \mathbf{W}(\mathbf{z} - \mathbf{x}) \\ \text{subject to} \quad & \mathbf{Gz} + \mathbf{h} \leq \mathbf{0} \end{aligned}$$

Mesh Simulation Flow: Backpropagation

Gradient computation available?

1. $\text{Init } \mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$ ✓ Handled by auto-differentiation
2. $\text{Compute } \Delta\mathbf{v} \text{ from } \mathbf{x}_t, \mathbf{v_t}$
   - $\Delta\mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$?
   - Newton's method
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta\mathbf{v}$ ✓ Handled by auto-differentiation
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$?
5. $t = t + 1, \text{ goto } 2$ ✓ Handled by auto-differentiation

Mesh Simulation Flow: Backpropagation

Gradient computation available?

1. Init $\mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$ ✓
2. Compute $\Delta \mathbf{v}$ from $\mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$ ✓ Using implicit differentiation!
   - Newton's method
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$ ✓
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$?
5. $t = t + 1,$ goto 2 ✓

Implicit Differentiation: Linear Solve

- Formulation: $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$
- Input: $\hat{\mathbf{M}}$ and $\mathbf{f}$. Output: $\mathbf{a}$
- Back propagation: use $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$ to compute $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$
  - $\mathcal{L}$: the loss function.

Implicit Differentiation: Linear Solve

- Back propagation: use $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$ to compute $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$, where $\mathcal{L}$ is the loss function.
- Implicit differentiation form: $\hat{\mathbf{M}}\partial\mathbf{a} = \partial\mathbf{f} - \partial\hat{\mathbf{M}}\mathbf{a}$
- Solution: $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d_a}\mathbf{z}^\top \quad \frac{\partial \mathcal{L}}{\partial \mathbf{f}} = \mathbf{d_a}^\top$

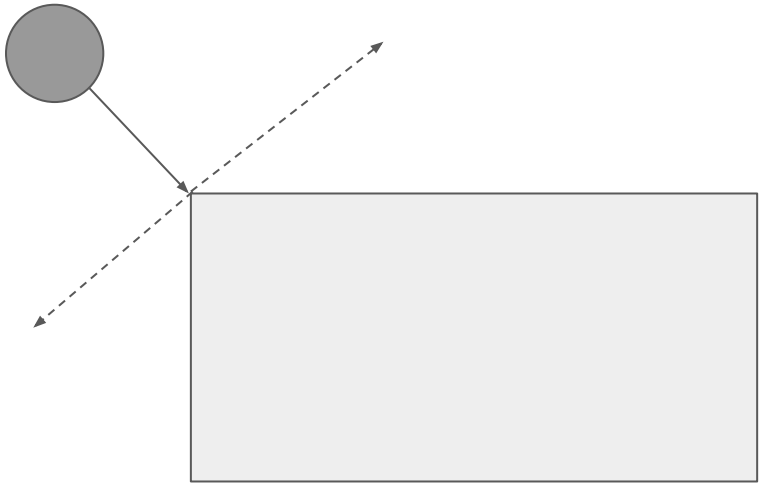
  where $\mathbf{d_a}$ is computed from $\hat{\mathbf{M}}^\top \mathbf{d_a} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}}^\top$, and $\mathbf{z}$ is the solution of $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$.

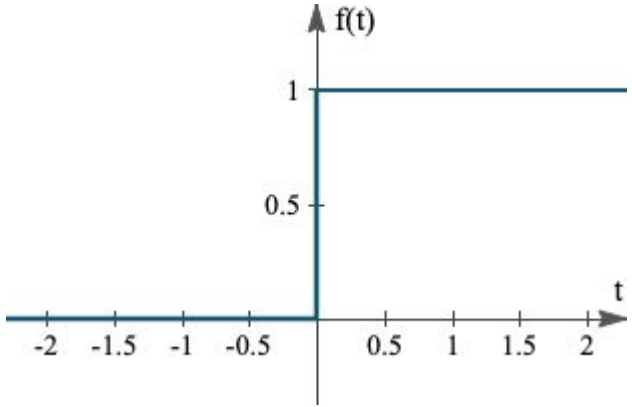
Mesh Simulation Flow: Backpropagation

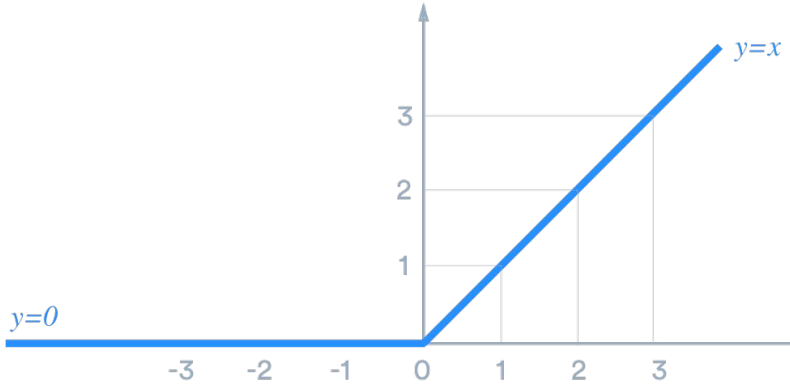
Gradient computation available?

1. Init $\mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$ ✓
2. Compute $\Delta \mathbf{v}$ from $\mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$ ✓
   - Newton's method
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$ ✓
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$ ✓ Using implicit differentiation!
5. $t = t + 1,$ goto 2 ✓

Gradients of Collision?







Collision Handling

- Objective formulation: Quadratic Programming

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{z}-\mathbf{x})^\top \mathbf{W}(\mathbf{z}-\mathbf{x}) \\ \text{subject to} \quad & \mathbf{G}\mathbf{z}+\mathbf{h} \leq \mathbf{0} \end{aligned}$$

z: optimized vertex positions
W: weight matrix
G, **h**: constraint matrices

Gradients of Collision Response

- Karush-Kuhn-Tucker (KKT) condition:

$$\begin{aligned}\mathbf{W}\mathbf{z}^* - \mathbf{W}\mathbf{x} + \mathbf{G}^\top \lambda^* &= 0 \\ D(\lambda^*)(\mathbf{G}\mathbf{z}^* + \mathbf{h}) &= 0\end{aligned}$$

- Implicit differentiation:

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \partial \mathbf{z} \\ \partial \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M}\partial \mathbf{x} - \partial \mathbf{G}^\top \lambda^* \\ -D(\lambda^*)(\partial \mathbf{G}\mathbf{z}^* + \partial \mathbf{h}) \end{bmatrix}$$

z: current vertex positions
W: weight matrix
G, **h**: constraint matrices
λ: Augmented Lagrangian multiplier
D(): diagonalize operator
*: optimization output

Gradients of Collision Response

- Solution:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{d}_{\mathbf{z}}^T \mathbf{W}$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{G}} = -D(\lambda^*)\mathbf{d}_\lambda \mathbf{z}^{*\top} - \lambda^* \mathbf{d}_{\mathbf{z}}^\top$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}} = -\mathbf{d}_\lambda^T D(\lambda^*).$$

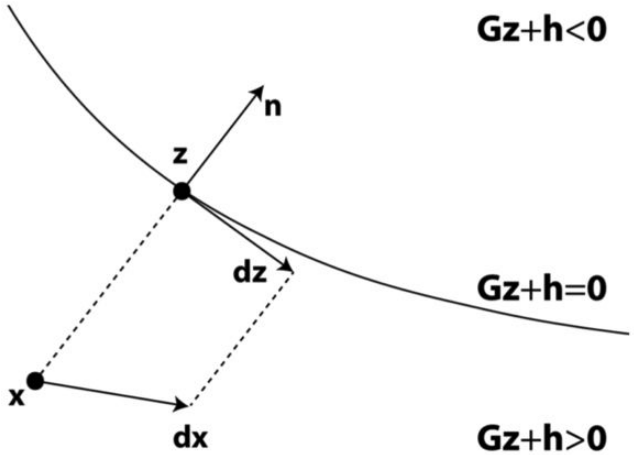
- where dz and dλ is provided by the linear equation:

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top D(\lambda^*) \\ \mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \mathbf{d}_\mathbf{z} \\ \mathbf{d}_\lambda \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top \\ \mathbf{0} \end{bmatrix}$$

Acceleration of Gradient Computation

- $$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top D(\lambda^*) \\ \mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \mathbf{d_z} \\ \mathbf{d}_\lambda \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top \\ \mathbf{0} \end{bmatrix}$$

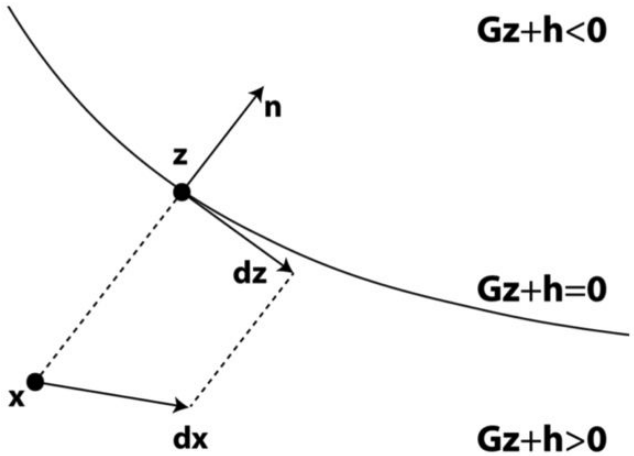
- Linear system of n+m
  - n: DOFs in the impacts
  - m: number of constraints/impacts
- Insight: Optimized point moves along the tangential direction w.r.t. constraint gradient



Acceleration of Gradient Computation

- $$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top D(\lambda^*) \\ \mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \mathbf{d_z} \\ \mathbf{d}_\lambda \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top \\ \mathbf{0} \end{bmatrix}$$

- Linear system of n+m
  - n: DOFs in the impacts
  - m: number of constraints/impacts
- Insight: Optimized point moves along the tangential direction w.r.t. constraint gradient



Acceleration of Gradient Computation

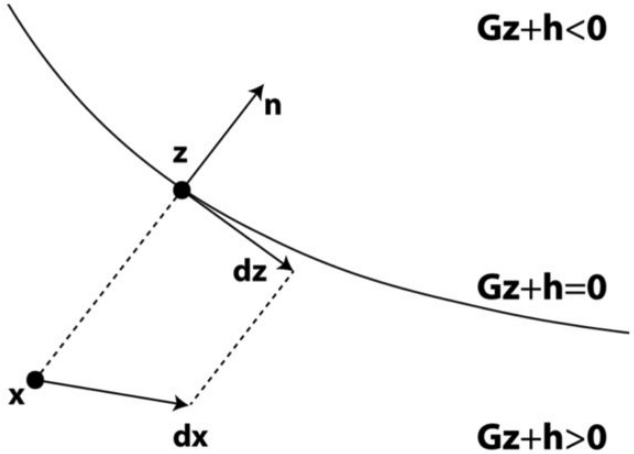
- Explicit solution of the linear equation:

$$\mathbf{d_z} = \sqrt{\mathbf{W}}^{-1}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\sqrt{\mathbf{W}}^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top$$

$$\mathbf{d}_\lambda = D(\lambda^*)^{-1}\mathbf{R}^{-1}\mathbf{Q}^\top\sqrt{\mathbf{W}}^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top$$

where Q and R is obtained from:

$$\sqrt{\mathbf{W}}^{-1}\mathbf{G}^\top = \mathbf{Q}\mathbf{R}$$



- Theoretical speedup: $O((n+m)^3) \rightarrow O(nm^2)$
  - n: number of vertices
  - m: number of constraints

Content

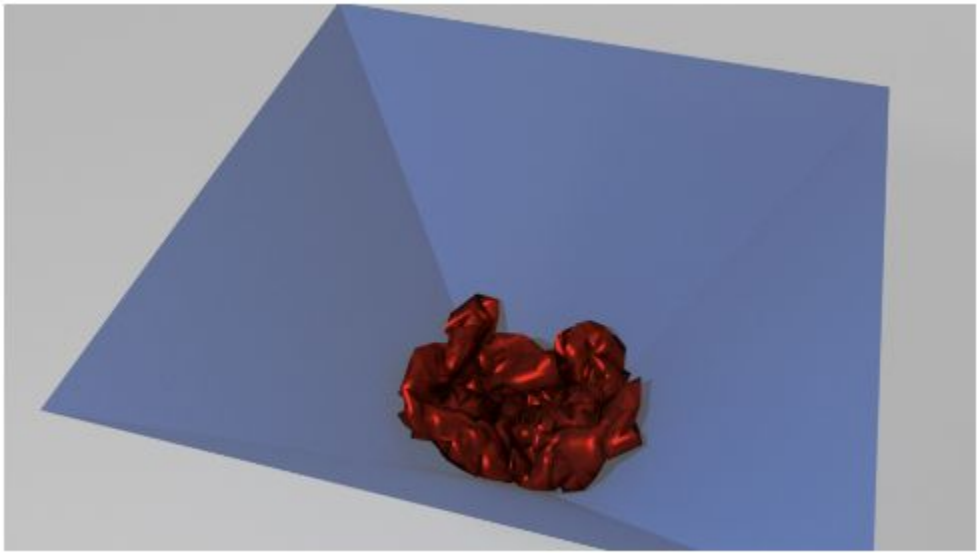
- Motivation
- Related Work
- Our Method
  - Simulation pipeline
  - Gradient Computation
- Results

Experimental Results

- Ablation study
  - Backpropagation speedup
- Applications
  - Material estimation
  - Motion control

Ablation Study

- Speed improvement in backpropagation
- Scene setting: a large piece of cloth crumpled inside a pyramid



Results

- Speed improvement in backpropagation
- Scene setting: a large piece of cloth crumpled inside a pyramid

Mesh resolution	Baseline		Ours		Speedup	
	Matrix size	Runtime (s)	Matrix size	Runtime (s)	Matrix size	Runtime
16x16	599 ± 76	0.33 ± 0.13	$\mathbf{66 \pm 26}$	$\mathbf{0.013 \pm 0.0019}$	8.9	25
32x32	1326 ± 23	1.2 ± 0.10	$\mathbf{97 \pm 24}$	$\mathbf{0.011 \pm 0.0023}$	13	112
64x64	2024 ± 274	4.6 ± 0.33	$\mathbf{242 \pm 47}$	$\mathbf{0.072 \pm 0.011}$	8.3	64

The runtime performance of gradient computation is significantly improved by up to two orders of magnitude.

Material Estimation

- Scene setting: A piece of cloth hanging under gravity and a constant wind force.

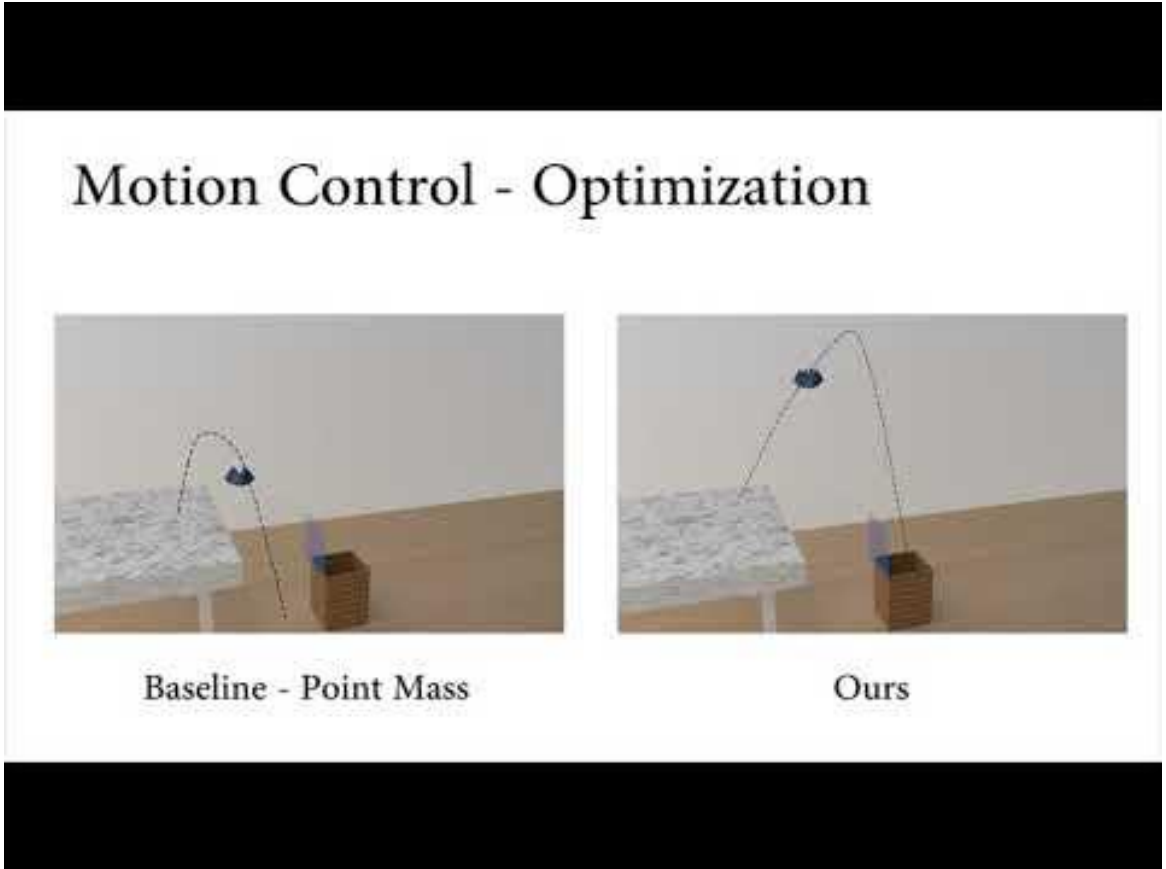
Results

- Application: Material estimation
- Scene setting: A piece of cloth hanging under gravity and a constant wind force.

Method	Runtime (sec/step/iter)	Density Error (%)	Non-Ln Streching Stiffness Error (%)	Ln Streching Stiffness Error (%)	Bending Stiffness Error (%)	Simulation Error (%)
Baseline	-	68 ± 46	74 ± 23	160 ± 119	$\mathbf{70 \pm 42}$	12 ± 3.0
L-BFGS [30]	2.89 ± 0.02	4.2 ± 5.6	64 ± 34	72 ± 90	$\mathbf{70 \pm 43}$	4.9 ± 3.3
Ours	$\mathbf{2.03 \pm 0.06}$	$\mathbf{1.8 \pm 2.0}$	$\mathbf{57 \pm 29}$	$\mathbf{45 \pm 41}$	77 ± 36	$\mathbf{1.6 \pm 1.4}$

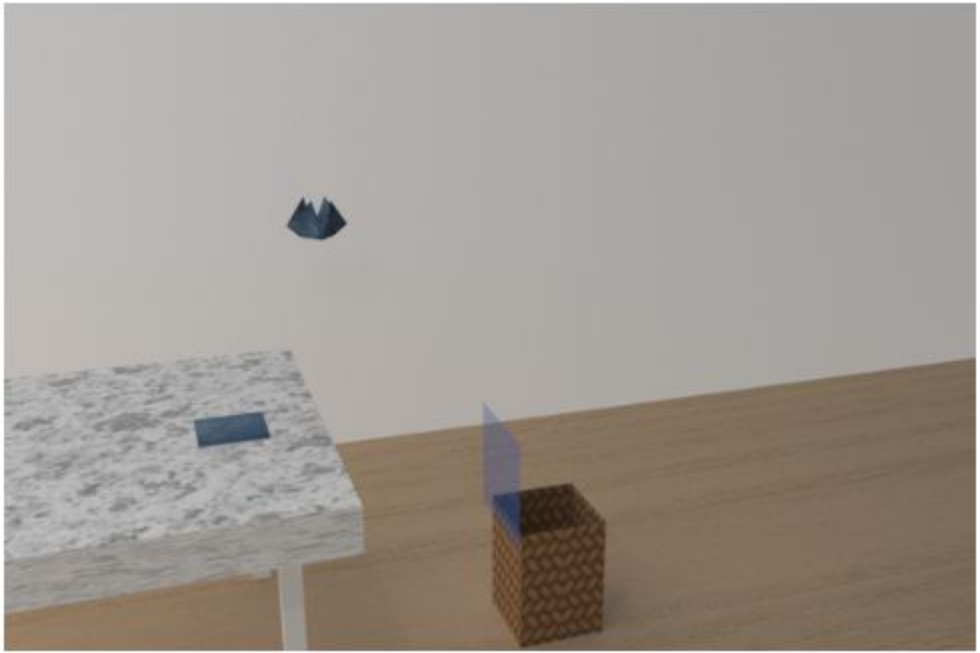
Our method achieves the fastest speed and the smallest overall error.

Application: Material Estimation



Motion Control

- Scene setting: A piece of cloth being lifted and dropped to a basket.



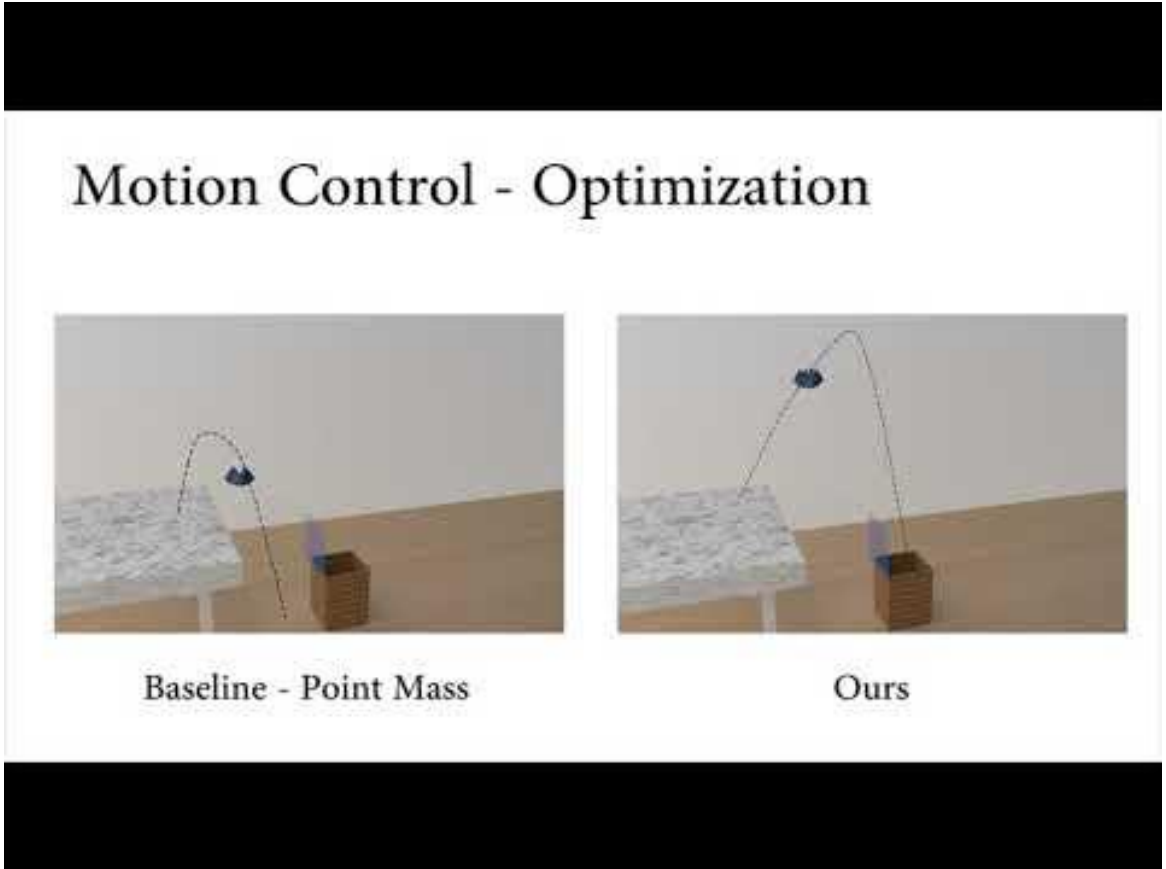
Results

- Application: Motion control
- Scene setting: A piece of cloth being lifted and dropped to a basket.

Method	Error (%)	Samples
Point Mass	111	–
PPO [18]	432	10,000
Ours	**17**	**53**
Ours+FC	39	108

Our method achieves the best performance with a much smaller number of simulations.

Application: Motion Control



Conclusion

- Differentiable simulation
  - Applicable to optimization tasks
  - Embedded in neural networks for learning and control
- Fast backpropagation for collision response

Future Work

- Optimization of the computation graph
  - Vectorization
  - PyTorch3D/DiffTaichi
- Integrate with other materials
  - Rigid body, deformable body, articulated body, etc

Q&A