Differentiable Physics for Learning & Control

Yi-Ling Qiao[1], Junbang Liang[1], Vladlen Koltun[2], and Ming Lin[1]

[1]University of Maryland at College Park

[2]Intel Labs

2

[1] ***Differentiable Cloth Simulation for Inverse Problems***
Junbang Liang and Ming C. Lin and Vladlen Koltun, NeurIPS 2019

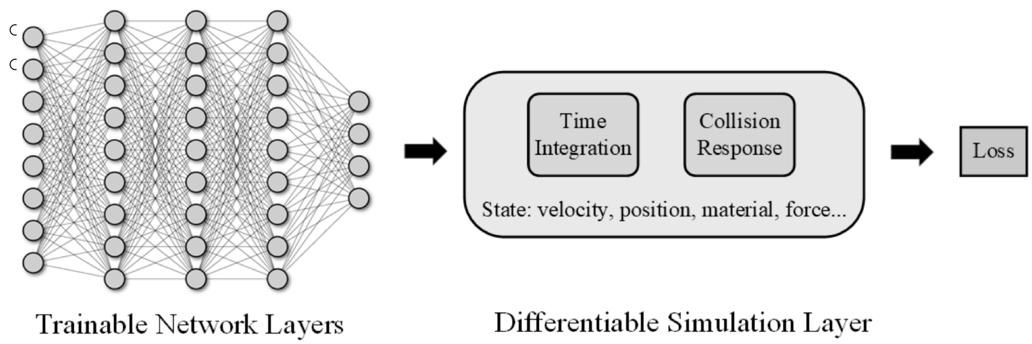
[2] ***Scalable Differentiable Physics for Learning and Control***
Yiling Qiao Junbang Liang, Vladlen Koltun, and Ming C. Lin, ICML 2020

[3] ***Efficient Differentiable Articulated Body Dynamics***
Yiling Qiao Junbang Liang, Vladlen Koltun, and Ming C. Lin, ICML 2021

[4] ***Differentiable Simulation of Soft Multi-Body Systems***
Yiling Qiao Junbang Liang, Vladlen Koltun, and Ming C. Lin, 2021

Motivation

- Differentiable Physics Simulation as a Network Layer
- Enables gradient-based learning and control
  - Material estimation, motion control, model-based reinforcement learning



4

Differentiable Cloth Simulation

Junbang Liang[1], Ming Lin[1], and Vladlen Koltun[2]

[1]University of Maryland at College Park

[2]Intel Labs

https://gamma.umd.edu/researchdirections/virtualtryon/differentiablecloth

UNIVERSITY OF MARYLAND **NeurIPS 2019** intel

5

Limitations with State-of-the-Art

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

- Learning-based physics
✓ *Unable to guarantee physical correctness*

6

Key Contributions

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

7

Gradients of Physics 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 $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$, where $\mathcal{L}$ is the loss function.
- 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 $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$.

8

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:

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

9

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:

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \mathbf{dz} \\ \mathbf{d}\lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{dx} - \mathbf{dG}^\top \lambda^* \\ -D(\lambda^*)(\mathbf{dG}\mathbf{z}^* + \mathbf{dh}) \end{bmatrix}$$

10

Gradients of Collision Response

- Solution:

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

where d_z and d_λ is provided by the linear equation:

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

11

Acceleration of Gradient Computation

- Explicit solution of the linear equation:

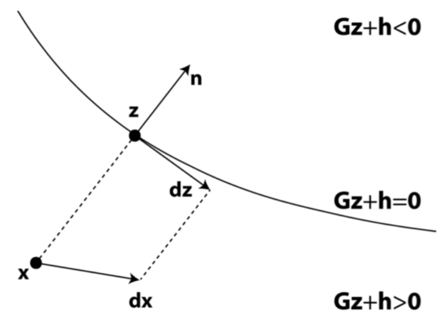
$$\mathbf{d_z} = \sqrt{\mathbf{W}}^{-1}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\sqrt{\mathbf{W}}^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top$$

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

where Q and R is obtained from:

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

- Theoretical speedup: $O((n+m)^3)$ → $O(nm^2)$



12

Results

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

Mesh Resolution	Baseline Matrix Size	Baseline Run Time (s)	Ours Matrix Size	Ours Run Time (s)	Speedup Matrix Size	Speedup Run Time
16x16	599 ± 76	0.33 ± 0.13	**66 ± 26**	**0.013 ± 0.0019**	8.9	25
32x32	1326 ± 23	1.2 ± 0.10	**97 ± 24**	**0.011 ± 0.0023**	13	112
64x64	2024 ± 274	4.6 ± 0.33	**242 ± 47**	**0.072 ± 0.011**	8.3	64

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

13

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 best runtime performance & the smallest error

14

Results

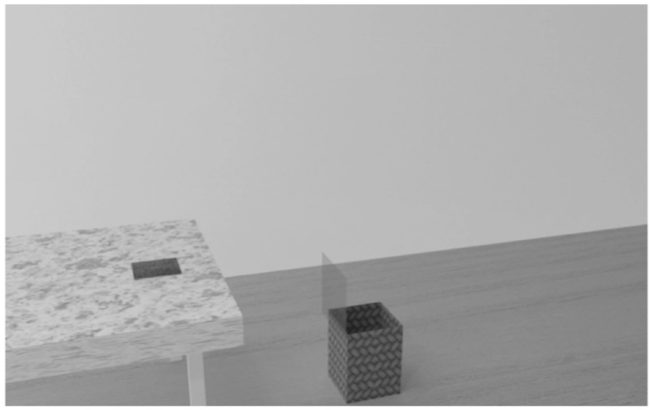
- Application: Motion control
- Scene setting: A piece of cloth being lifted and dropped to a basket.

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

15

Video Demos



Baseline - Treating as point mass

16

Summary

- A fully differentiable cloth simulation
  - Dynamic collision handling
  - Derivations of gradients using implicit differentiation
- Backpropagation acceleration by using QR decomposition to obtain the explicit solution
- Application examples: material estimation and motion control
  - Enabling ‘simulate-and-compare’ when embedding with deep network

17

Scalable Differentiable Physics for Learning and Control

Yi-Ling Qiao[1], Junbang Liang[1], Vladlen Koltun[2], and **Ming C. Lin[1]**

[1]University of Maryland at College Park

[2]Intel Labs

https://gamma.umd.edu/researchdirections/mlphysics/diffsim/

UNIVERSITY OF MARYLAND

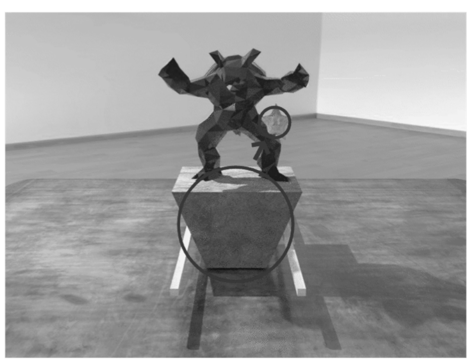
ICML 2020

intel

18

Motivation

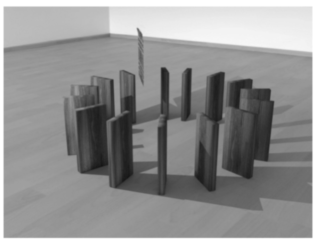
- Differentiable Physics Simulation as a Network Layer
  - Control of physical systems

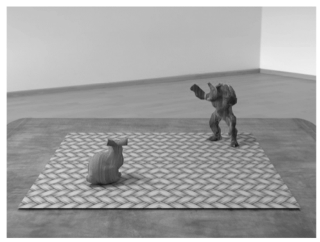


19

Motivation

- **Scalable** Differentiable Physics
  - Large **number** of interacting objects
  - Non-trivial **shapes**
  - Large variety of object **sizes**
  - Different physical properties/**material types**





20

Related Work

- Particle based differentiable simulation
  - DiffTaichi (Hu et al. 2019)
  - Cannot scale to large scenes: cubic growth regarding resolution/sizes
- Rigid body differentiable simulation
  - Degrave et al. (2017) (collisions only between balls and planes)
  - de Avila Belbute-Peres et al. (2018) (2D Simulator)
  - Not general enough: cannot support general 3D shapes
- Mesh based differentiable cloth simulation
  - Liang et al. (2019)
  - Not general enough: 3D deformable cloth only

21

Related Work

- Particle based differentiable simulation
  - DiffTaichi (Hu et al. 2019)
  - Cannot scale to large scenes: cubic growth regarding resolution/sizes
- Rigid body differentiable simulation
  - Degrave et al. (2017) (collisions only between balls and planes)
  - de Avila Belbute-Peres et al. (2018) (2D Simulator)
  - Not general enough: cannot support general 3D shapes
- Mesh based differentiable cloth simulation
  - Liang et al. (2019)
  - Not general enough: 3D deformable cloth only

22

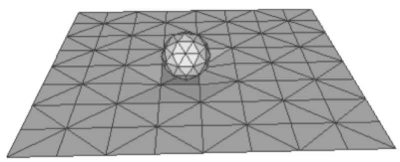
Related Work

- Particle based differentiable simulation
  - DiffTaichi (Hu et al. 2019)
  - Cannot scale to large scenes: cubic growth regarding resolution/sizes
- Rigid body differentiable simulation
  - Degrave et al. (2017) (collisions only between balls and planes)
  - de Avila Belbute-Peres et al. (2018) (2D Simulator)
  - Not general enough: cannot support general 3D shapes
- Mesh based differentiable cloth simulation
  - Liang et al. (2019)
  - Not general enough: 3D deformable cloth only

23

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth



24

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth

25

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth

26

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth

27

Mesh Simulation Flow

1. $\text{Init } \mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$
2. $\text{Compute } \Delta \mathbf{v} \text{ from } \mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$
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. $t = t + 1, \text{ goto } 2$

28

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

29

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.

30

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: $\partial\hat{\mathbf{M}}\mathbf{a} + \hat{\mathbf{M}}\partial\mathbf{a} = \partial\mathbf{f}$
- Solution: $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d}_{\mathbf{a}}\mathbf{z}^\top \quad \frac{\partial \mathcal{L}}{\partial \mathbf{f}} = \mathbf{d}_{\mathbf{a}}^\top$,

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

31

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! Algorithm-dependent
5. $t = t + 1,\ \text{goto } 2$ ✓

32

Our Goal

- Scalability regarding resolution and shape
  - Mesh-based representation
- Scalability regarding material and quantity
  - Coupled physics between rigid body and deformable cloth
  - Localized collision handling

33

Mesh 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. $t = t + 1, \text{ goto } 2$

34

Dynamics Formulation

- Simulated objects: rigid body and deformable cloth
- Degree of freedom: 6 for rigid body, 3m for deformable cloth
- Stacked general coordinates: $\boldsymbol{q} = [\boldsymbol{q}_1^\top, \boldsymbol{q}_2^\top, ..., \boldsymbol{q}_n^\top]^\top$
    - $\boldsymbol{q}_k \in \mathbb{R}^6$ for rigid bodies
    - $\boldsymbol{q}_k \in \mathbb{R}^{3m_k}$ for clothes
- Dynamics:

$$\frac{d}{dt}\begin{pmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{pmatrix}$$

k: object index, m: # of vertices of a cloth, n: # of objects

35

Collision Handling

- Global LCP solve for rigid bodies
  - Good at static contacts and static frictions
  - Difficult to couple with other materials
  - Slow
- Local constraint solver for clothes
  - Impulse-based solution: easy to couple between different materials
  - Solve within independent zones: faster computation
  - Unstable for large scale static contacts

36

Collision Handling

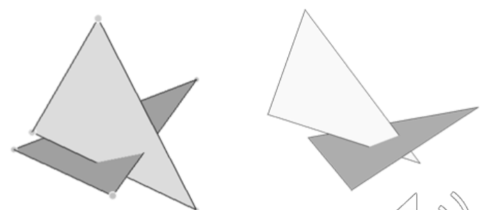
- Global LCP solve for rigid bodies
  - Good at static contacts and static frictions
  - Difficult to couple with other materials
  - Slow
- Local constraint solver for clothes
  - Impulse-based solution: easy to couple between different materials
  - Solve within independent zones: faster computation
  - Unstable for large scale static contacts

37

Local Collision Handling

Impact zone model (Harmon et al. 2008)

- Constraints built upon impacts
- Linear w.r.t. vertex positions
  - $C_{ee} = \boldsymbol{n} \cdot [(\alpha_3 \boldsymbol{x}_3 + \alpha_4 \boldsymbol{x}_4) - (\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2)]$
    $C_{vf} = \boldsymbol{n} \cdot [\boldsymbol{x}_4 - (\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2 + \alpha_3 \boldsymbol{x}_3)]$



EE Impact　　VF Imp

α_i: barycentric coordinates of each vertex at collision

Local Collision Handling

Impact zone model (Harmon et al. 2008)

- Introduce minimum energy: QP formulation

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

composed of: $C_{vf} = n \cdot [x_4 - (\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3)] < 0$

- Solve the optimization for each `connected component' of impacts

$\mathbf{M}$: mass matrix, $\mathbf{G}$ and $\mathbf{h}$: constraint matrix and vector

Local Collision Handling

Coupling with rigid bodies

- Treating one rigid body as one node
- Nonlinear constraint for optimization

$$\begin{aligned} \underset{\mathbf{q}'}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{q}-\mathbf{q}')^\top \hat{\mathbf{M}}(\mathbf{q}-\mathbf{q}') \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{f}(\mathbf{q}')+\mathbf{h} \leq \mathbf{0} \end{aligned}$$

$\mathbf{f}(\cdot)$: a function mapping from general coordinates q to some endpoint $\mathbf{x}$

Gradient Computation for Collision

- Optimization problem:

$$\begin{aligned} \underset{\mathbf{q}'}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{q}-\mathbf{q}')^\top \hat{\mathbf{M}}(\mathbf{q}-\mathbf{q}') \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{f}(\mathbf{q}')+\mathbf{h} \leq \mathbf{0} \end{aligned}$$

- Karush-Kuhn-Tucker (KKT) condition at optimal point:
  - Stationarity (gradient=0), and complementary slackness

$$\begin{aligned} \hat{\mathbf{M}}\mathbf{z}^* - \hat{\mathbf{M}}\mathbf{q} + \nabla \boldsymbol{f}^\top \mathbf{G}^\top \lambda^* &= 0 \\ D(\lambda^*)(\mathbf{G}\boldsymbol{f}(\mathbf{z}^*)+\mathbf{h}) &= 0 \end{aligned}$$

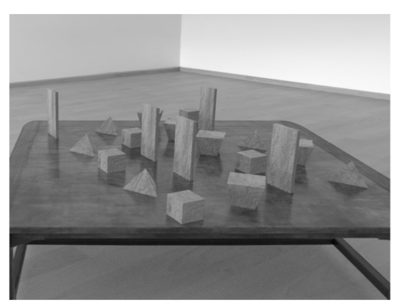
Results

1. Scalable
   - Large number of objects - Linear w.r.t. number of objects
   - High resolution
2. General
   - Complex objects
   - Two-way Coupling - Constant
3. Applications
   - Inverse Problem - Faster than derivative-free methods
   - Control - Faster than RL

46

Results - Scalable

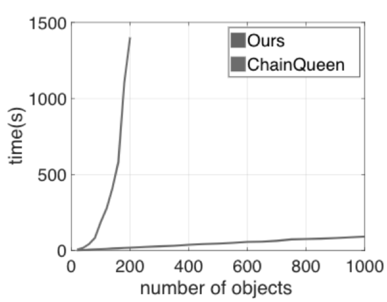
- Scale the number of objects
- Scene setting: A bunch of (20 - 1000) objects collide with the ground.
  - Methods: Ours vs. ChainQueen[8] (on CPU, for 2 second)
  - Scale the number of objects, while keeping the density of collisions and objects
  - When the number of object scales from 20 to 200, the grid size of ChainQueen[8] scales from 64 to 640

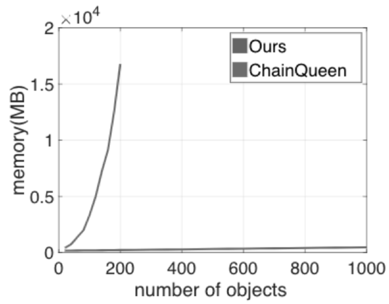


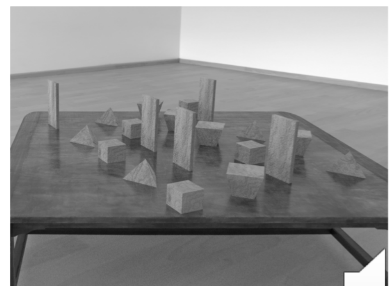
47

Results - Scalable

- Scale the number of objects
- Scene setting: A bunch of (20 - 1000) objects collide with the ground.
- **Our method scales well (linearly) in large scenes with big number of objects.**



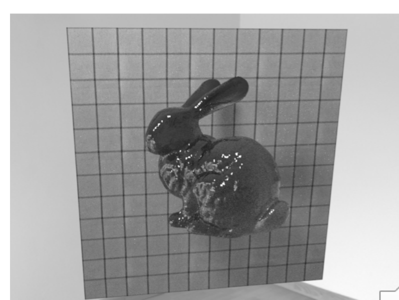




48

Results - Scalable

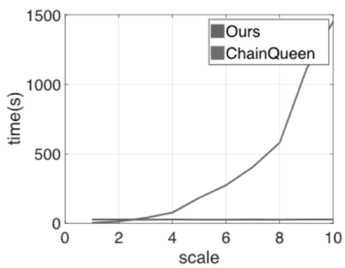
- Scale the resolution
- Scene setting: A bunny and a piece of cloth. Vary the relative sizes of cloth.
  - Methods: Ours vs. ChainQueen[8] (on CPU, for 2 second)
  - The relative size of two cloths: n:1.
  - n scales from 1 to 10.
  - The grid size of ChainQueen[8] scales from 64 to 640



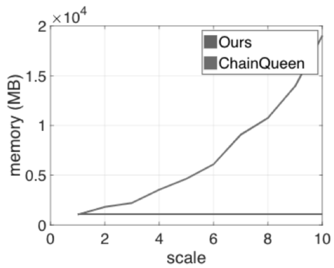
49

Results - Scalable

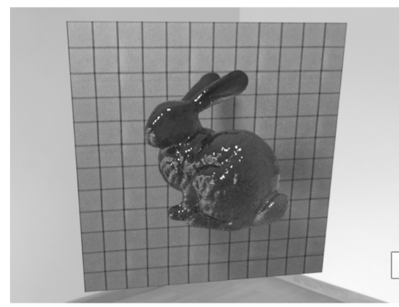
- Scale the resolution
- Scene setting: A bunny and two piece of cloths. Vary the relative sizes of clothes.
- **Our method runs in constant time in different resolutions.**



(b) Running time



(c) Memory consumption



50

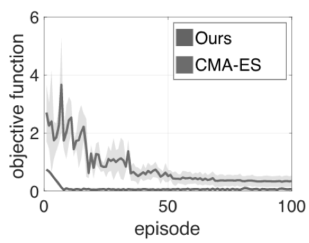
Results - Inverse Problem

- Learn the trajectory
- Scene setting: Compute the force on the marble in each step to drive it to a target point.
  - Methods: Ours vs. CMA-ES
  - The combined force vector has 100 dimensions
  - Object function: the distance to target + norm of the force vector

51

Results - Inverse Problem

- Learn the trajectory
- Scene setting: Compute the force on the marble in each step to drive it to a target point.
- **Our method converges more than 10x faster than CMA-ES.**



(b) Objective function

52

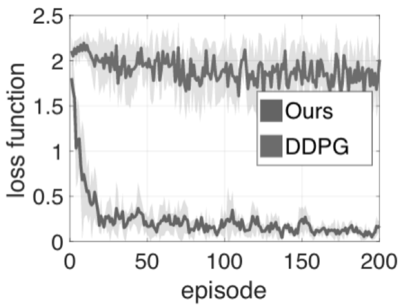
Results - Control

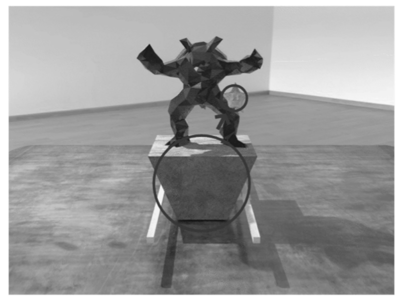
- Manipulation
- Scene setting: Control the motion of a pair of parallel grippers, to move an object towards a random target in 2 second
  - Methods: Ours vs. DDPG[6]
  - Fixed initial position and random target.
  - Loss is the L2 distance from the target to the current position. Reward = -1 * Loss
  - Observation: [x_now - x_target, v_now, time, reward]
  - Action: [v_next]

53

Results - Control

- Manipulation
- Scene setting: Control the motion of a pair of parallel grippers, to move an object towards a random target
- **Our method converges much faster than RL**





54

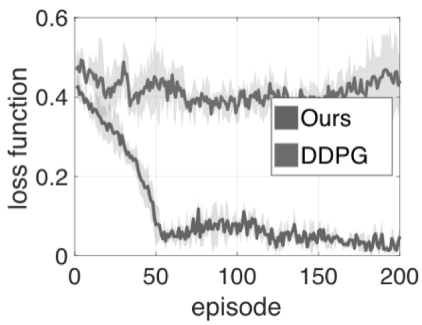
Results - Control

- Motion control
- Scene setting: Control the motion of four handles on a cloth, to move the cube towards a random target
  - Methods: Ours, DDPG[6]
  - Fixed initial position and random target.
  - Loss is the L2 distance from the target to the current position. Reward = -1 * Loss
  - Observation: [x_now - x_target, v_now, time, reward]
  - Action: [v_next]

55

Results - Control

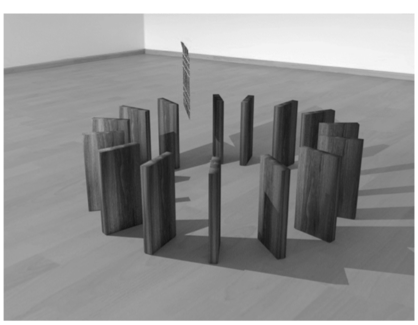
- Motion control
- Scene setting: Control the motion of four handles on a cloth, to move the cube towards a random target
- **Our method converges much faster than RL**

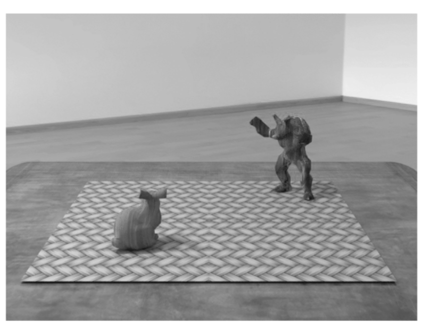


56

Results - General

- Two-way coupling between cloth and rigid body
- Scene setting: Cloth & dominos

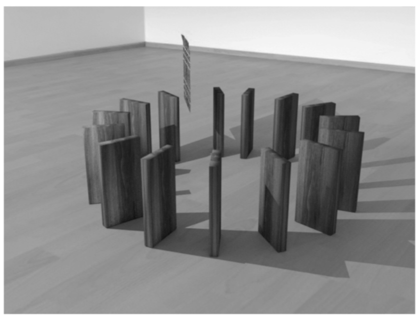


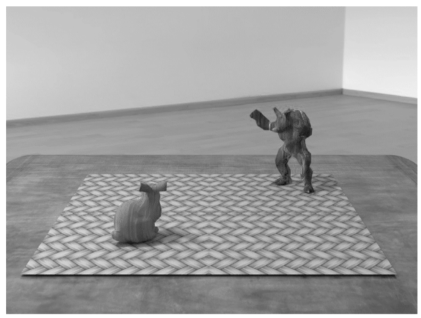


57

Conclusion

- A method for scalable and general differentiable physics
- Future work
  - More general dynamics
  - More Application





58

Video Demonstration

Scalable Differentiable Physics for
Learning and Control

Submission ID: 15

59

Efficient Differentiable Articulated Dynamics

Yi-Ling Qiao*, Junbang Liang*, Vladlen Koltun, and Ming Lin*

* UNIVERSITY OF MARYLAND

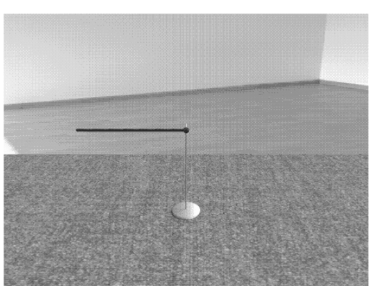
intel

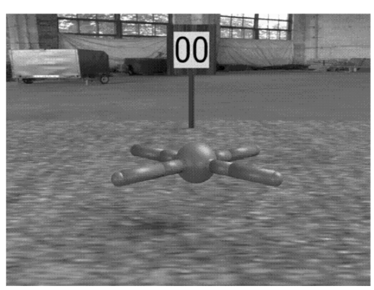
Code & data: https://github.com/YilingQiao/diffarticulated

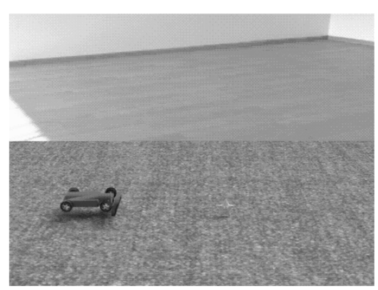
60

Motivation

- Differentiable articulated body simulation as a network layer
  - Control physical systems
  - Enhance reinforcement learning
  - Estimate physics parameters

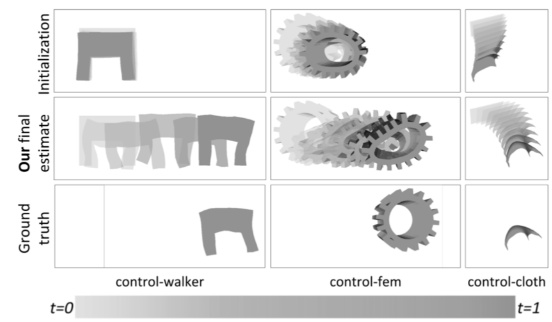




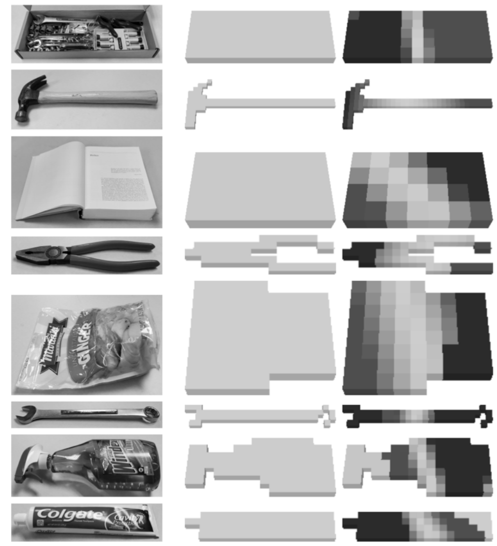


61

Applications



[2] Murthy et al. (2021)

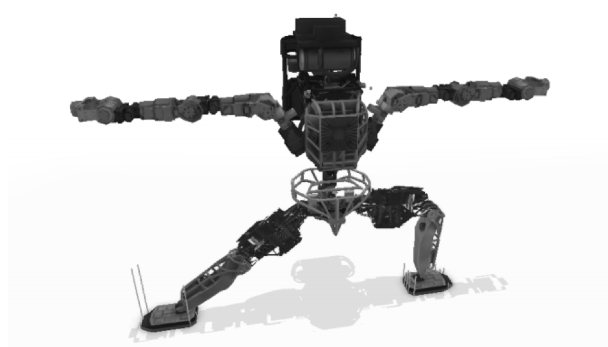


[5] Song et al. (2020)

62

Related/concurrent work

[1] Geilinger et al. (2020)



[7] Werling et al. (2021)

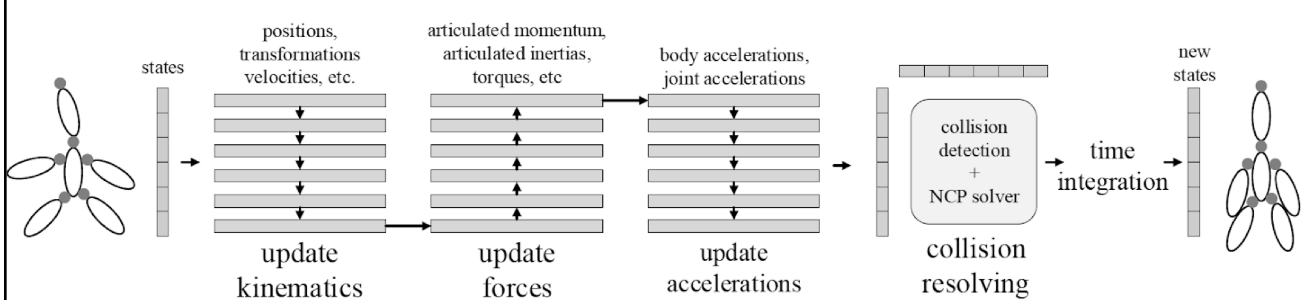
63

Content

- Related Work
- Our Method
  - Differentiating the simulation
  - Application to reinforcement learning
- Results

64

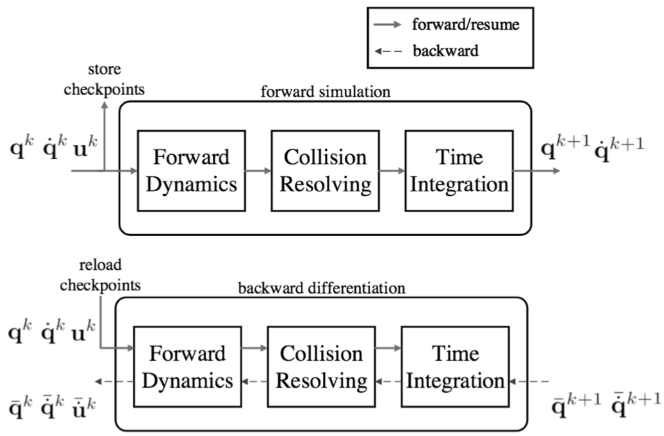
Workflow of one simulation step



65

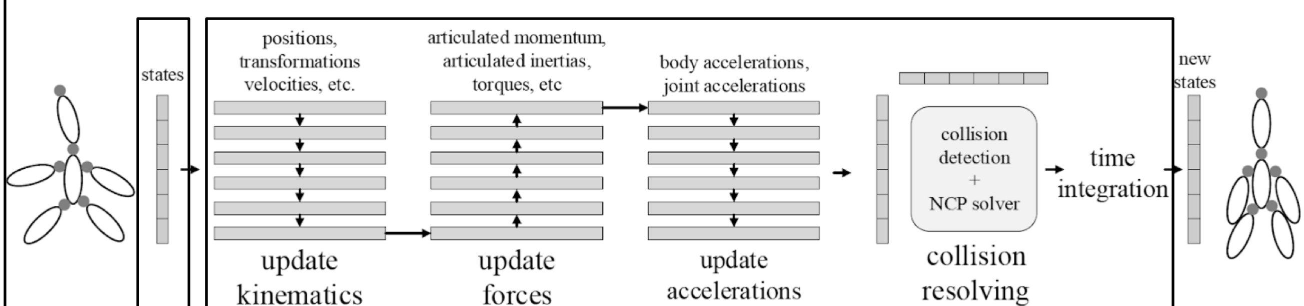
Checkpointing

Forward and backward workflow with checkpointing scheme



66

Checkpointing



checkpoints

Intermediate variables

67

Application with Reinforcement Learning

- Sample enhancement
  - Increase sample efficiency
  - Faster convergence
- Policy enhancement
  - Update the policy using analytic gradients
  - Better scalability in high dimensionality

68

Sample Enhancement

- Idea: Use simulation gradients to generate extra nearby examples
- Point sample → patch sample
  - Faster convergence

$$a_k = a_0 + \Delta a_k$$
$$s'_k = s'_0 + \boxed{\frac{\partial s'_0}{\partial a_0}} \Delta a_k$$
$$r_k = r_0 + \boxed{\frac{\partial r_0}{\partial a_0}} \Delta a_k$$

Enabled by differentiable simulation!

a: action
s: observation
s': next-step observation
r: reward

69

Policy Enhancement

- Idea: Use simulation gradients to compute better policy gradients
- Use one-step rollout to approximate the action gradients

$$\mathcal{L}_\mu = -Q(s, \mu(s)) + Z$$

Soft Actor-Critic

$$\frac{\partial Q(s,a)}{\partial a} = \boxed{\frac{\partial r}{\partial a}} + \gamma \frac{\partial Q(s', \mu(s'))}{\partial s'} \boxed{\frac{\partial s'}{\partial a}}$$

$$\mathcal{L}'_\mu = -\frac{\partial Q(s,a)}{\partial a} \mu(s) + Z$$

Ours

Enabled by differentiable simulation!

a: action
s: observation
s': next-step observation
r: reward
Q: critic network
μ: policy network
Z: regularization term

70

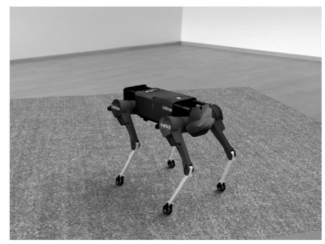
Content

- Related Work
- Our Method
  - Adjoint derivation
  - Application with reinforcement learning
- Results

71

Results - Performance

- Compare the runtime and memory usage.
- Scene: One Laikago released from the air and hitting the ground
  - Scale the simulation length: 50, 100, 500, 1000, 5000 steps
- Comparisons:
  - Use autodiff tools in the same simulation pipeline



72

Results - Performance

- Compare the runtime and memory usage.
- Scene: A Laikago released from the air and hitting the ground
- Our method has the highest speed and the lowest memory usage
  - x10 faster than autodiff tools with 1% of memory usage

steps	50	100	500	1000	5000
ADF	25.7	25.5	25.1	32.1	58.4
Ceres	27.2	27.5	27.2	34.0	58.2
CppAD	2.4	2.4	2.3	2.3	4.5
JAX	53.3	46.1	43.1	42.7	42.3
PyTorch	195.6	192.2	199.2	192.8	N/A
Ours	**0.3**	**0.3**	**0.2**	**0.2**	**0.2**

Forward simulation time (ms) per step

steps	50	100	500	1000	5000
ADF	25.7	25.5	25.1	32.1	58.4
Ceres	27.2	27.5	27.2	34.0	58.2
CppAD	2.4	2.4	2.3	2.3	4.5
JAX	53.3	46.1	43.1	42.7	42.3
PyTorch	195.6	192.2	199.2	192.8	N/A
Ours	**0.3**	**0.3**	**0.2**	**0.2**	**0.2**

Peak Memory (MB)

73

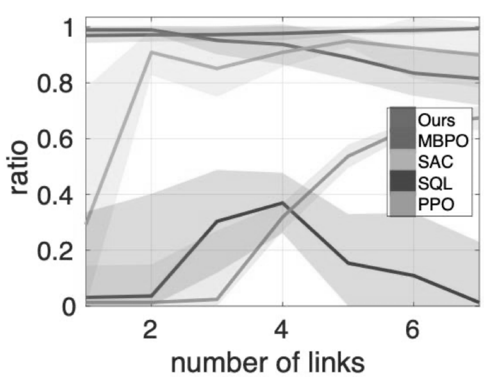
Policy Enhancement

- Scenario: N-link pendulum
- Objective: reaching the highest point within 100 frames
- Reward
  - -dist_to_target^2
- Baseline: MBPO, SAC, SQL, PPO
- Number of links: 1-7
- Number of training epochs: 100 * n_links
  - Samples per epoch: 100

74

Policy Enhancement

- Test metric: Best relative reward
  - Absolute reward / maximum possible reward (reaching exactly the target)

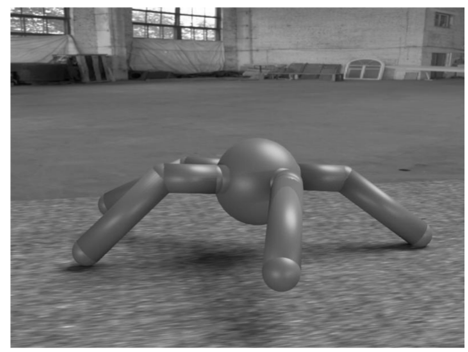


Our method scales with increasing system complexity

75

Sample Enhancement

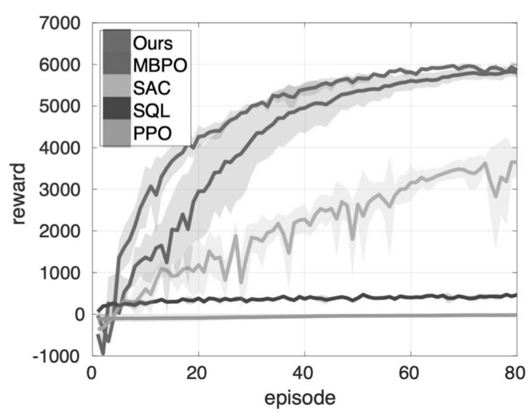
- Scenario: Mujoco Ant
- Objective: walking towards +x axis
- Reward
  - v_x - sum(action^2)
- Baseline: MBPO, SAC, SQL, PPO
- Number of training epochs: 100
  - Samples per epoch: 1000



76

Sample Enhancement

- Test metric:
  - Maximum (absolute) reward



Our method achieves the same best reward and converges faster

77

Video Demonstration

https://youtu.be/RrWGLfR4wfk

78

Differentiable Simulation of Soft Multi-body Systems

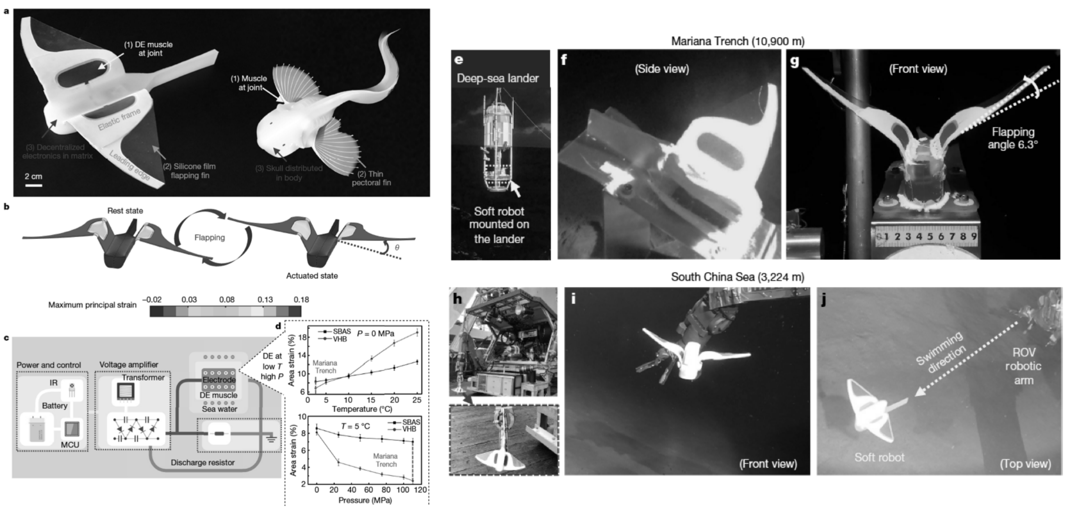
Yi-Ling Qiao*, Junbang Liang*, Vladlen Koltun, and Ming Lin*

* UNIVERSITY OF MARYLAND intel

79

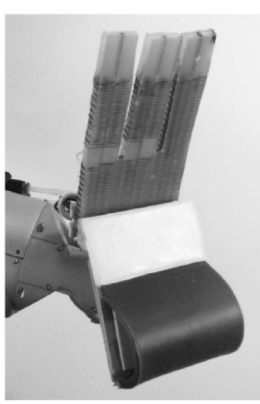
Motivation

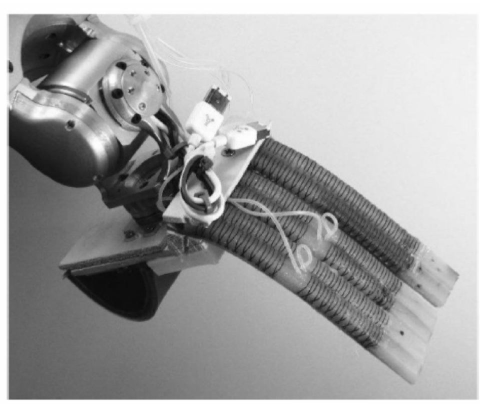
- Self-powered soft robot in the Mariana Trench



Motivation

- A Compliant Hand Based on a Novel Pneumatic Actuator.

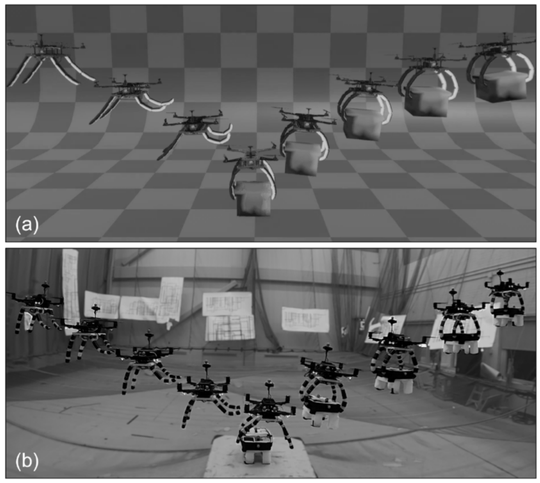




81

Motivation

- Dynamic Grasping with a "Soft" Drone



82

OBJECTIVE

- ***Differentiable Physics Simulator*** to support different scenarios
  - Complex Contact
  - Embedded Skeleton
  - Joint, muscle, and pneumatic actuators

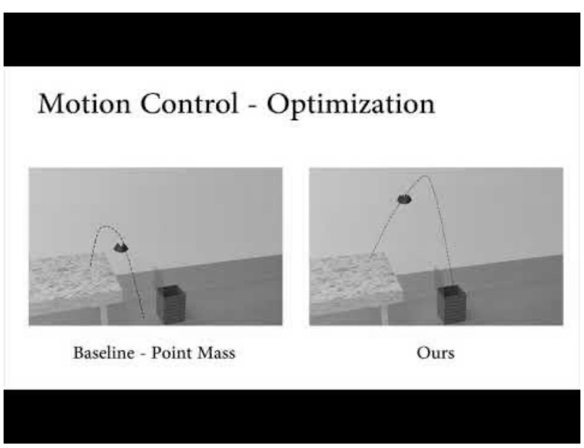
83

Content

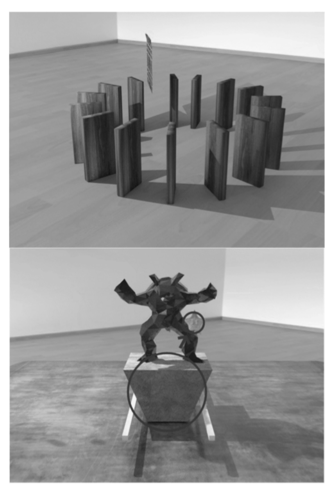
- Related Work
- Background
- Our Method
  - Articulation
  - Contact
- Results

84

Related Work



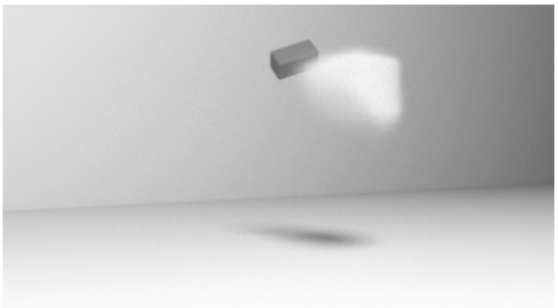
Liang et al. (NeurIPS 2019)
cloth



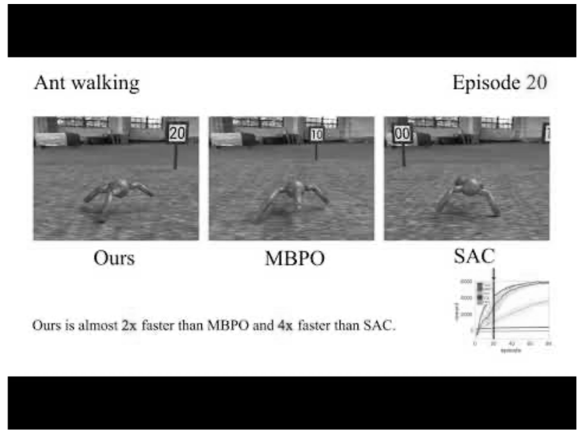
Qiao et al. (ICML 2020)
cloth + rigid body

85

Related Work



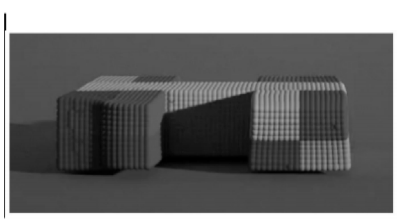
Takahashi et al. (AAAI 2021)
Fluids + rigid body



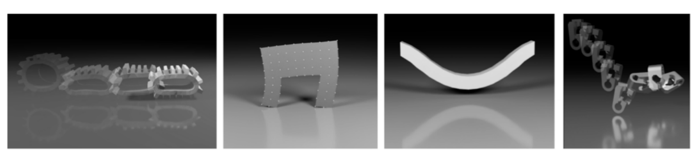
Qiao et al. (ICML 2021)
Articulated body

86

Related Work



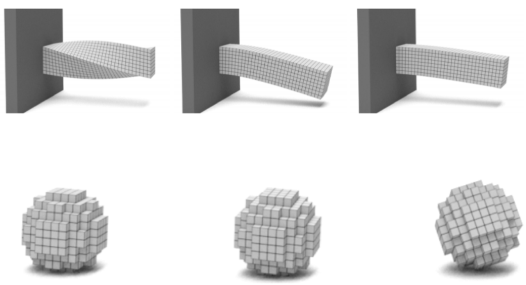
Difftaichi (2019)



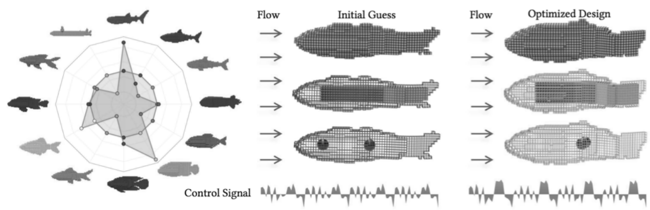
Gradsim (2021)

87

Related Work



DiffPD (2021)



DiffAqua (2021)

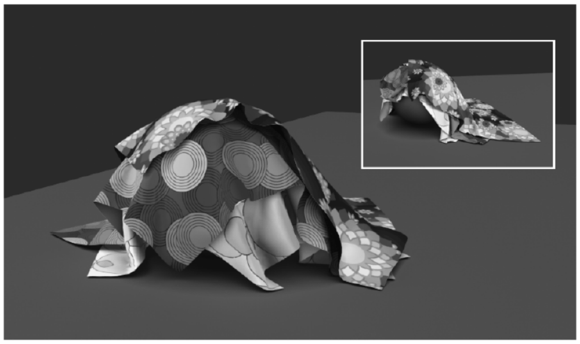
88

Content

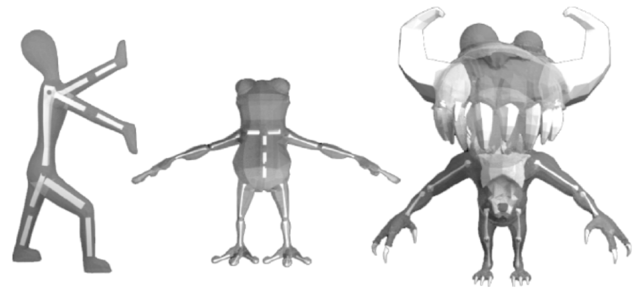
- Related Work
- Background
- Our Method
  - Articulation
  - Contact
- Results

89

Background



[6] Ly et al. (2020)



[1] Li et al. (2018)

90

Background

Projective dynamics

Implicit Euler : $$\mathbf{M}(\mathbf{q}_{n+1} - \mathbf{q}_n - h\mathbf{v}_n) = h^2(\nabla E(\mathbf{q}_{n+1}) + \mathbf{f}_{ext})$$

Solve: $$\mathbf{q}_{n+1} = \arg\min_{\mathbf{q}} \frac{1}{2h^2}(\mathbf{q} - \mathbf{s}_n)^\top \mathbf{M}(\mathbf{q} - \mathbf{s}_n) + E(\mathbf{q})$$

$$E(\mathbf{q}) = \sum_i \frac{\omega_i}{2} \|\mathbf{G}_i \mathbf{q} - \mathbf{p}_i\|_F^2$$

Local step:

Global step: $$\mathbf{q}_{n+1} = \arg\min_{\mathbf{q}} \frac{1}{2}\mathbf{q}^\top \left(\frac{\mathbf{M}}{h^2} + \mathbf{L}\right)\mathbf{q} + \mathbf{q}^\top \left(\frac{\mathbf{M}}{h^2}\mathbf{s}_n + \mathbf{J}\mathbf{p}\right)$$

91

Content

- Related Work
- Background
- Our Method
  - Articulation
  - Contact
- Results

Method - rigid bodies

Vertices on rigid bodies : $\mathbf{q}_k = \mathbf{Q}\mathbf{T}_k^r\mathbf{V}_k$

Linearize: $\mathbf{q}_k^{i+1} = \mathbf{q}_k^i + \Delta\mathbf{q}_k^i = \mathbf{q}_k^i + \frac{\partial \mathbf{q}_k^i}{\partial \mathbf{z}_k}\Delta\mathbf{z}_k^i$ $\quad\quad$ $\mathbf{B} = \frac{\partial \mathbf{q}^i}{\partial \mathbf{z}}$

New global step:

$$\Delta\mathbf{z}^i = \arg\min_{\Delta\mathbf{z}} \frac{1}{2}\Delta\mathbf{z}^\top\mathbf{B}^\top\left(\frac{\mathbf{M}}{h^2}+\mathbf{L}\right)\mathbf{B}\Delta\mathbf{z} + \Delta\mathbf{z}^\top\mathbf{B}^\top\left(\left(\frac{\mathbf{M}}{h^2}+\mathbf{L}\right)\mathbf{q}^i - \left(\frac{\mathbf{M}}{h^2}\mathbf{s}_n + \mathbf{J}\mathbf{p}\right)\right)$$

Local step:

$$\mathbf{T}_k = \begin{bmatrix}\mathbf{I}+\omega_k^{i*} & \mathbf{0}\\ \mathbf{0} & 1\end{bmatrix}\mathbf{T}_k^r + \begin{bmatrix}\mathbf{0} & \mathbf{l}_k^i\\ \mathbf{0} & 0\end{bmatrix} = \mathbf{U}\Sigma\mathbf{V}^\top$$

$$\mathbf{T}_k^{r\prime} = \mathbf{U}\mathbf{V}^\top$$

Method - Articulated body

Skeleton tree: $\mathbf{T}_k^r = \prod_u \mathbf{A}_u$

A is the local transformation matrix

Jacobian:

$$\mathbf{B}_{u,v} = \frac{\partial \mathbf{T}_u^r \mathbf{V}_u}{\partial \mathbf{z}_v} = \mathbf{Q}\mathbf{P}_v \frac{\partial \mathbf{A}_v}{\partial \mathbf{z}_v} \mathbf{S}_{v,u} \mathbf{V}_u$$

Compute recursively:

$$\mathbf{P}_v = \mathbf{P}_{v'} \mathbf{A}_{v'}$$
$$\mathbf{S}_{v',u} = \mathbf{A}_v \mathbf{S}_{v,u}$$

P is the prefix product
S is the suffix product

Method - Articulated body

Algorithm 2 Matrix Assembly for the Articulated System

Input: tree link u
Compute $\mathbf{P}_u$ using Eq. 16
$v \leftarrow u$
while v is not root **do**
    Compute $\mathbf{S}_{v,u}$ using Eq. 17
    Compute $\mathbf{B}_{u,v}$ using Eq. 13
    $v \leftarrow$ parent(v)
end while
Compute $\mathbf{B}_{u,root}$ using Eq. 15
for s **in** descendants(u) **do**
    Solve link s recursively
end for

Method - Articulated body

Rotational joint. This joint is characterized by a rotation axis $\mathbf{n}$ and the angle θ. Its transformation matrix and the Jacobian are:

$$\mathbf{A}^r = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \frac{\partial \mathbf{A}^r}{\partial \theta} = \begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \theta} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \tag{18}$$

$$\mathbf{R} = \cos\theta \cdot \mathbf{I} + \sin\theta [\mathbf{n}]_\times + (1 - \cos\theta)\mathbf{n}\mathbf{n}^\top \tag{19}$$

$$\frac{\partial \mathbf{R}}{\partial \theta} = -\sin\theta \cdot \mathbf{I} + \cos\theta [\mathbf{n}]_\times + \sin\theta \mathbf{n}\mathbf{n}^\top \tag{20}$$

The local update of the rotational joint is given by:

$$\theta^{i+1} = \arctan(\sin\theta^i + \cos\theta^i \Delta\theta^i, \cos\theta^i - \sin\theta^i \Delta\theta^i) \tag{21}$$

Method - Articulated body

Prismatic joint. This joint is characterized by a prismatic axis $\mathbf{u}$ and the scale l. Its transformation matrix and the Jacobian are:

$$\mathbf{A}^p = \begin{bmatrix} \mathbf{I} & l\mathbf{u} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \frac{\partial \mathbf{A}^p}{\partial l} = \begin{bmatrix} \mathbf{0} & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix} \tag{22}$$

(23)

The local update of the prismatic joint is simply addition:

$$l^{i+1} = l^i + \Delta l^i \tag{24}$$

Method - Actuation - Joint Torque

$$\Delta\mathbf{z}^i = \arg\min_{\Delta\mathbf{z}} \frac{1}{2}\Delta\mathbf{z}^\top\mathbf{B}^\top\left(\frac{\mathbf{M}}{h^2}+\mathbf{L}\right)\mathbf{B}\Delta\mathbf{z} + \Delta\mathbf{z}^\top\mathbf{B}^\top\left(\left(\frac{\mathbf{M}}{h^2}+\mathbf{L}\right)\mathbf{q}^i - \left(\frac{\mathbf{M}}{h^2}\mathbf{s}_n + \mathbf{J}\mathbf{p}\right)\right) \quad (8)$$

Solve a linear system: $\begin{bmatrix}\mathbf{H}_d & \mathbf{H}_c^\top \\ \mathbf{H}_c & \mathbf{H}_r\end{bmatrix}\begin{bmatrix}\Delta\mathbf{z}_d^i \\ \Delta\mathbf{z}_r^i\end{bmatrix} = \begin{bmatrix}\mathbf{k}_d \\ \mathbf{k}_r\end{bmatrix}$

Torques can be added to K_r directly

98

Method - Actuation - Pneumatic

Pneumatic actuator. We use co-rotational elastic strain energy model for tetrahedral cells. For a pneumatic cell with activation level a, the energy is computed as

$$\Psi_{pneumatic}(\mathbf{F}, a) = \frac{k_p}{2}\left\|\mathbf{F} - \mathbf{R}(a)\right\|^2 \quad (27)$$

where the SVD decomposition of the deformation gradient is $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{R}(a) = \mathbf{U}\Sigma^*\mathbf{V}^T$, $\Sigma^* = \mathbf{D} + \Sigma$, and $\mathbf{D}$ is computed by

$$\arg\min_{\mathbf{D}} \|\mathbf{D}\|_2^2, s.t. \prod_i(\Sigma_i i + \mathbf{D}_i) = a \quad (28)$$

99

Method - Actuation - Muscle

Muscle actuator. We use the muscle actuators described in [49]. Muscles are modeled as fibers in the soft bodies, and the forces are computed as $\mathbf{f}_{muscle}(a) = -f_{muscle}(a)\mathbf{m}$, where $a \in [0,1]$ is the activation level, $\mathbf{m}$ is the direction of fiber. To achieve this force, a strain energy model [32] is used, $E_{muscle} = \mathbf{V}_{muscle}\Psi_{muscle}(\mathbf{F}, e)$, where $\Psi_{muscle}(\mathbf{F}, a) = \frac{k_m}{2}\|(1-r)\mathbf{F}\mathbf{m}\|$, k_m is the stiffness, $r = \frac{1-a}{l}$ is the projection of the cord segment, $l = \|\mathbf{F}\mathbf{m}\|$ is the stretch factor.

100

Content

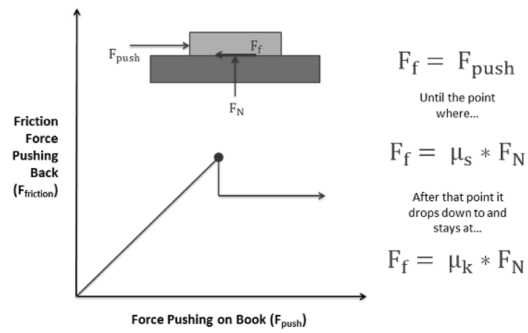
- Related Work
- Background
- Our Method
  - Articulation
  - Contact
- Results

101

Method - Contact

- Dry friction contact model
  - Coulomb's friction law
- Simplification

  $\mu_s = \mu_k$
- Implementation
  - Normal momentum: cancelled out
  - Tangential momentum
    - Reverse impulse proportional to the normal momentum
    - No larger than the current momentum



102

Method - Contact

Original global step:

$$\left(\frac{\mathbf{M}}{h^2} + \mathbf{L}\right) \mathbf{q}_{n+1} = \frac{\mathbf{M}}{h^2}\mathbf{s}_n + \mathbf{J}\mathbf{p}$$

Convert to velocity space:

$$\overbrace{\mathbf{M}\mathbf{v}^{i+1}}^{\text{Adjusted momentum}} = \overbrace{\mathbf{f} - h^2\mathbf{L}\mathbf{v}^i}^{\text{Current momentum}} + \xi^i$$

$$\mathbf{f} = \mathbf{M}\mathbf{s}_n - (\mathbf{M} + h^2\mathbf{L})\mathbf{q}_n + h^2\mathbf{J}\mathbf{p}$$

Contact handling:

ξ^i Depends on the relative velocities/momentums of collided vertices

103

Method - Contact

- Friction law enforcement
  - The new impulse is added to the individual vertex
  - Iteratively resolved until converged
- Convergence
  - Not guaranteed
  - Depends on **M** and **L** if **f** and ξ are fixed
- Applicability to soft bodies
  - **L** too large compared to **M**
  - Unstable solve

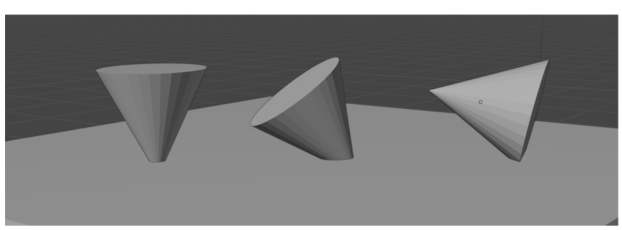
$$\overbrace{\mathbf{M}\mathbf{v}^{i+1}}^{\text{Adjusted momentum}} = \overbrace{\mathbf{f} - h^2\mathbf{L}\mathbf{v}^i}^{\text{Current momentum}} + \xi^i$$

$$\mathbf{f} = \mathbf{M}\mathbf{s}_n - (\mathbf{M} + h^2\mathbf{L})\mathbf{q}_n + h^2\mathbf{J}\mathbf{p}$$

104

Method - Contact

- Improvement
  - Move the diagonals of **L** to the left!
  - $(\mathbf{M} + h^2\mathbf{D})\mathbf{v}^{i+1} = \mathbf{f} - h^2(\mathbf{L} - \mathbf{D})\mathbf{v}^i + \xi^i$
  - When **f** and ξ are fixed, the improved method is guaranteed to converge
- Contact detection
  - Continuous collision detection
  - Grouped vertex-face collision handling
    - Contact forces need to be computed jointly



105

Content

- Related Work
- Background
- Our Method
  - Checkpoint method
  - Adjoint derivation
  - Application with reinforcement learning
- Results

106

Results

- Implementation
- Ablation study
- Parameter estimation
- Motion Control

107

Results - Implementation

- Differentiation: Autodiff + Eigen3 + Checkpointing scheme

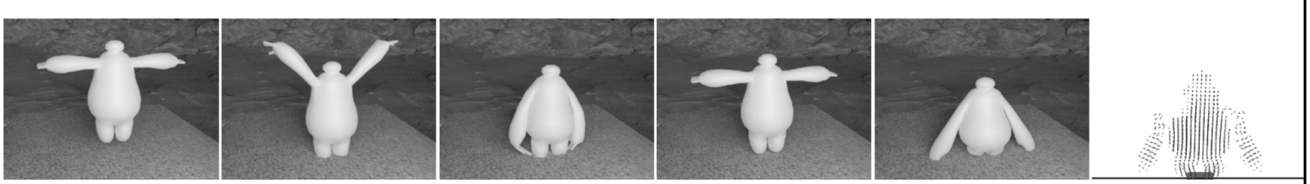
Table 1: Memory usage (GB).

steps	w/o ckpt	w/ ckpt
10	0.9	0.1
20	1.4	0.1
100	6.9	0.1
200	15.7	0.1

108

Results - Ablation Study

- Skeleton

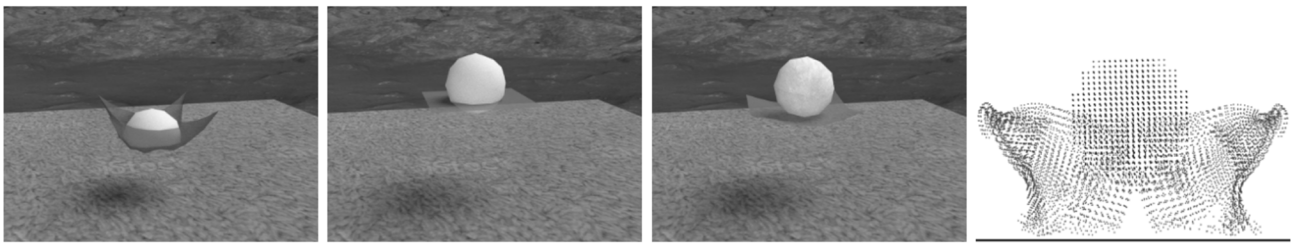


(a) Start (b) Ours (c) Passive [36] (d) Diffsim [50] (e) No skeleton (f) MPM[22]

109

Results - Ablation Study

- Contact

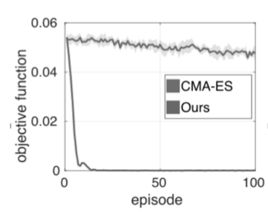


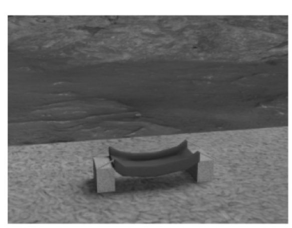
(a) Ours (b) Projective [45] (c) Diffsim [50] (d) MPM [22]

110

Parameter Estimation

- Scenario: suspension bridge
- Optimization variable: Young's modulus and Poisson's ratio
- Objective: Compliance under gravity
- Baseline: CMAES

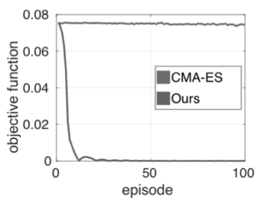


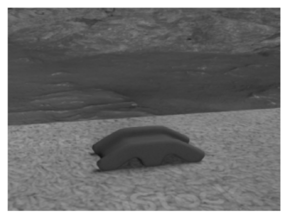


111

Parameter Estimation

- Scenario: arch bridge
- Optimization variable: Young's modulus and Poisson's ratio
- Objective: Compliance under gravity
- Baseline: CMAES

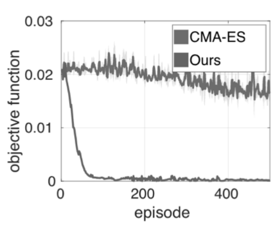


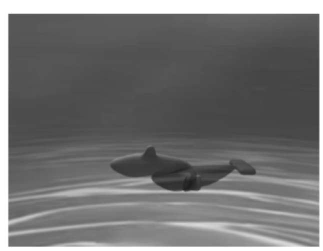


112

Motion Control - Skeleton

- Scenario: control a fish
- Optimization variable: joint torque
- Objective: reach a target place
- Baseline: CMAES

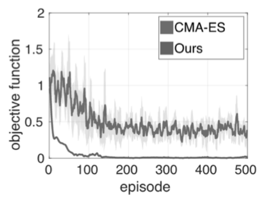


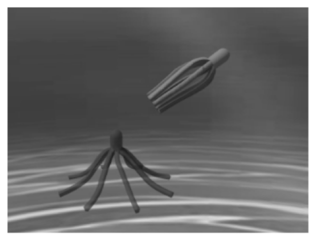


113

Motion Control - Muscle

- Scenario: control an octopus
- Optimization variable: muscle actuation levels
- Objective: reach a target place
- Baseline: CMAES

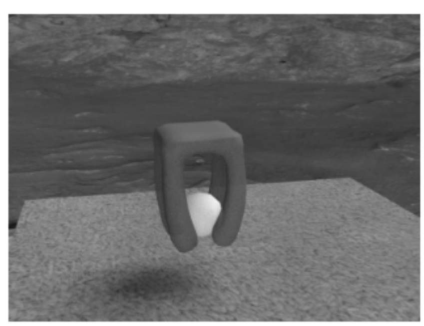




114

Motion Control - Pneumatic

- Scenario: control a gripper
- Optimization variable: pneumatic activation
- Objective: reach a target place
- Baseline: CMAES/MBPO



115

Video Demonstration

https://youtu.be/TPgFM5WxzaU

116

Questions?

https://gamma.umd.edu/researchdirections/virtualtryon/differentiablecloth

https://gamma.umd.edu/researchdirections/mlphysics/diffsim/

lin@cs.umd.edu

THANK YOU!!!

117

Reference

[1] *Differentiable Cloth Simulation for Inverse Problems*
Junbang Liang and Ming C. Lin and Vladlen Koltun, NeurIPS 2019

[2] *Scalable Differentiable Physics for Learning and Control*
Yiling Qiao Junbang Liang, Vladlen Koltun, and Ming C. Lin, ICML 2020

[3] *Efficient Differentiable Articulated Body Dynamics*
Yiling Qiao Junbang Liang, Vladlen Koltun, and Ming C. Lin, ICML 2021

[4] *Differentiable Simulation of Soft Multi-Body Systems*
Yiling Qiao Junbang Liang, Vladlen Koltun, and Ming C. Lin, 2021

[5] **Differentiable fluids with solid coupling for learning and control.**
Takahashi, T., Liang, J., Qiao, Y.-L., and Lin, M. C. AAAI 2021

Additional References

[1] DiffTaichi: Differentiable Programming for Physical Simulation
Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand

[2] **Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids** *Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba*

[3] **A Differentiable Physics Engine for Deep Learning in Robotics**
Jonas Degrave, Michiel Hermans, Joni Dambre, and Francis wyffels

[4] **End-to-End Differentiable Physics for Learning and Control**
de Avila Belbute-Peres F, Smith K, Allen K, Tenenbaum J, and Kolter JZ

[5] **Continuous control with deep reinforcement learning**
Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra

Additional References

[6] **ADD: analytically differentiable dynamics for multi-body systems with frictional contact.**
Geilinger, M., Hahn, D., Zehnder, J., Bacher, M., Thomaszewski, ¨ B., and Coros, S.
ACM TOG 2020.

[7] **Gradsim: Differentiable simulation for system identification and visuomotor control.**
Murthy, J. K., Macklin, M., Golemo, F., Voleti, V., Petrini, L.,Weiss, M., Considine, B., Parent-L´evesque, J., Xie, K., Erleben,K., Paull, L., Shkurti, F., Nowrouzezahrai, D., and Fidler, S.
ICLR 2021.

[8] **Learning to slide unknown objects with differentiable physics simulations.**
Song, C. and Boularias, A. Robotics: Science and Systems (RSS) 2020.

[9] **Fast and Feature-Complete Differentiable Physics for Articulated Rigid Bodies with Contact.**
Keenon Werling, D. Omens, J. Lee, I. Exarchos and C. K. Liu. RSS 2021.

120